27/5/2015

م.أحمد جابر

9

Memory Interface

- Memory are made up of (**registers**).
- Each register consists of one storage location
- Each location consists of an **address**
- The number of storage locations from few hundreds to several mega or giga locations
- The total number of memory storage is called memory **capacity** and measured in Bytes
- Each register consists of storage element (FF, capacitor for semiconductor)
- A storage element is called **cell**
- The data could be read from or written to memory

What is a memory?

A memory is a device that stores information in electrical, magnetic or optical form. Memories are used for storage of both program and data .

What are bit and byte organized memories?

For a $2^n \times m$ memory, **where n is the number of address lines** such that a maximum of 2^n memory locations can be accessed and **m is the word length**, if m is 1 then the memory is said to be **bit** organized, whereas if m is 8, then the memory is said to be **byte** organized.

Example

A semiconductor memory is specified as 4 K × 8. Indicate the number of words, word size and total capacity of this memory.

Total number of words or memory locations that can be accessed is

$= 4\ K = 4 \times 1024 = 4096$

Word size = 8

and total capacity of the memory is

$= 4096 \times 8 = 32768$

$= 32{,}768$ bits

- ***Semiconductor memories*** have become very popular and widely used because of their ***high reliability, low cost, high speed*** and ***ease with which memory size can be expanded.***

- ***It can be categorised into two ways:***
  1- Primary memory or main memory or working memory.
  2- Secondary memory or auxiliary memory or mass storage.

- RAM and ROM comprise the primary memory while magnetic tapes, magnetic disks, floppy disks or compact disks (CDs) are examples of secondary memory.

Comparison between primary and secondary memory

S. No.	Primary memory		Secondary memory
1.	Can store less amount of data.	1.	Can store huge amount of data.
2.	Faster speed of operation.	2.	Slower speed of operation.
3.	Can be volatile/non-volatile in nature.	3.	Always non-volatile in nature.
4.	Program/data used by the programmer are resident in the main memory.	4.	Not used for such purpose.
5.	Can be directly accessed by CPU.	5.	Cannot be directly accessed by CPU but can be accessed through I/O ports or in a serial format using hardware/software.
6.	If the CPU has n address lines, a maximum of 2^n main memory locations can be accessed.	6.	No such relation exists.
7.	Less costly.	7.	Costlier than main memory.

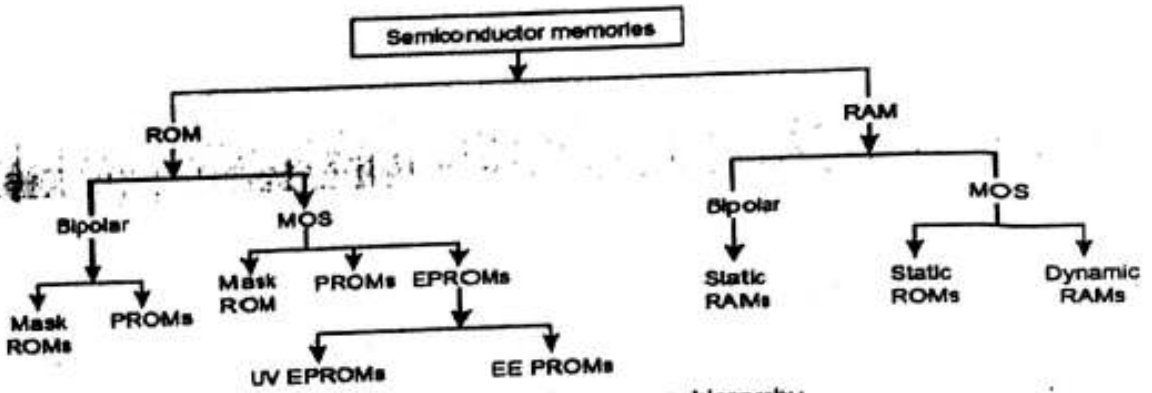
- The **memory map** is a guide showing how the entire system memory has been allocated to ROM, RAM so that any future memory expansions can be executed effectively.

- Memories are made of storage elements, which can store one bit of data. Each storage element is called a cell. In semiconductor memories, **flip-flops** act as storage elements.

- At each memory location, which is identified by its address, one or more number of bits can be stored. The number of bits that a particular memory location can store is known as working length of a memory. This also goes by the name of **word size.**

Different versions of semiconductor memories available.

Basically semiconductor memories can be RAM (Random Access Memory) or ROM (Read Only Memory)-both of which are available in bipolar technology or MOS (Metal Oxide Semiconductor) versions. The different versions available are:



Semiconductor memory hierarchy

What is a EPROM? Mention its two types and compare.

EPROMs are manufactured with NMOSFET technology with an isolated gate structure. There are two types of EPROM. These are:

UVEPROM : **Ultraviolet erasable programmable Read-only-Memory**
EEPROM : **Electrically erasable programmable Read-only-Memory (also known as EAPROM)**

Comparison between mask programmable ROM and PROM

S. No.	Mask programmable ROM		PROM
1.	Manufactured by making special masks.	1.	Manufactured by blowing fusible nichrome wire links.
2.	Programmed at the factory premises.	2.	Programmed by the user.
3.	No reprogramming possible.	3.	No reprogramming possible.
4.	Less costly.	4.	More costly.

Comparison between EEPROM and UVPROM

S. No.	EEPROM		UVPROM
1.	Can be erased and programmed with electrical signals.	1.	Can be erased and programmed with ultraviolet light.
2.	The voltage on the floating gate structure allows storage of information.	2.	The photo current from the insulated gate structure allows storage of information.
3.	Higher speed of operation.	3.	Lower speed of operation.
4.	More expensive.	4.	Less expensive.
5.	Relatively easy to manufacture.	5.	Relatively difficult to manufacture.
6.	Erasing takes several minutes.	6.	Erasing takes several minutes.
7.	Less packing density.	7.	More packing density.

3

Memory Types

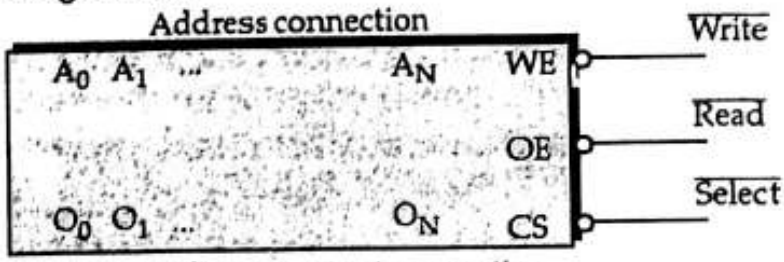
Two basic types:

- **ROM**: Read-only memory
- **RAM**: Read-Write memory

Four commonly used memories:

- ROM
- Flash (EEPROM)
- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Generic pin configuration:



Memory Chips

The number of address pins is related to the number of *memory locations*.

Common sizes today are **1K** to **256M** locations.

Therefore, between 10 and 28 address pins are present.

The data pins are typically *bi-directional* in read-write memories.

The number of data pins is related to the size of the *memory location*.

For example, an 8-bit wide (byte-wide) memory device has 8 data pins.

Catalog listing of *1K X 8* indicate a byte addressable 8K memory.

Each memory device has at least one *chip select* ($\overline{CS}$) or *chip enable* ($\overline{CE}$) or *select* ($\overline{S}$) pin that enables the memory device.

This enables read and/or write operations.

If more than one are present, then all must be 0 in order to perform a read or write.

Memory Chips

Each memory device has at least one control pin.

For ROMs, an *output enable* ($\overline{OE}$) or *gate* ($\overline{G}$) is present.

The $\overline{OE}$ pin enables and disables a set of tristate buffers.

For RAMs, a *read-write* (R/$\overline{W}$) or *write enable* ($\overline{WE}$) and *read enable* ($\overline{OE}$) are present.

For dual control pin devices, it must be hold true that both are not 0 at the same time.

ROM:

Non-volatile memory: Maintains its state when powered down.

There are several forms:

- **ROM**: Factory programmed, cannot be changed. Older style.
- **PROM**: Programmable Read-Only Memory.
  Field programmable but only once. Older style.
- **EPROM**: Erasable Programmable Read-Only Memory.
  Reprogramming requires up to 20 minutes of high-intensity UV light exposure.

Memory Chips

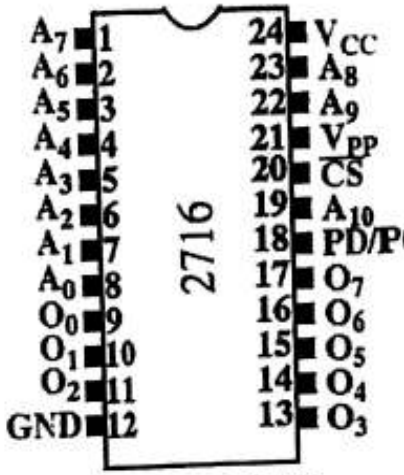
ROMs (cont):

- **Flash EEPROM**: Electrically Erasable Programmable ROM.
  Also called **EAROM** (Electrically Alterable ROM) and **NOVRAM** (NOn-Volatile RAM).
  Writing is much slower than a normal RAM.

  Used to store setup information, e.g. video card, on computer systems.
  Can be used to replace EPROM for BIOS memory.

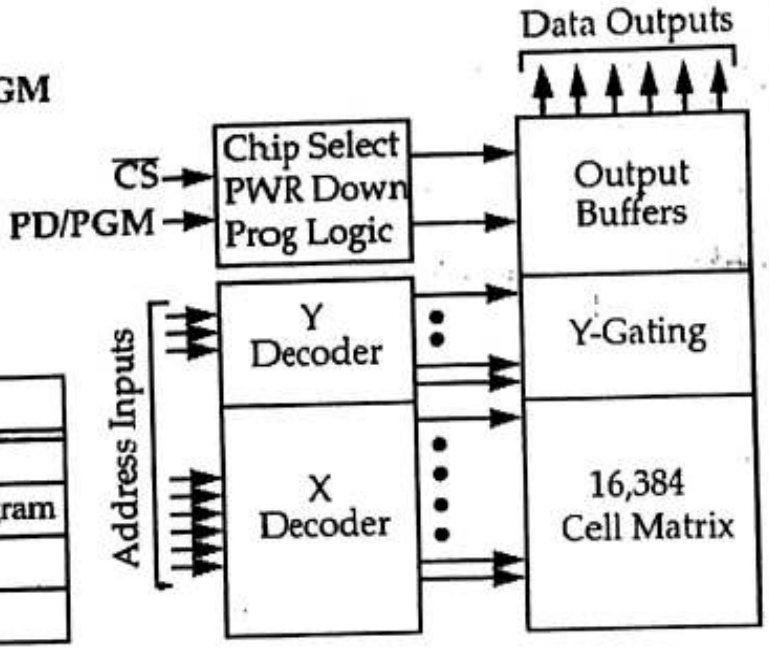
EPROMs

Intel 2716 EPROM (2K X 8):



2K x 8 EPROM

V_{PP} is used to program the device by applying 25V and pulsing PGM while holding $\overline{CS}$ high.



Pin(s)	Function
A_0-A_{10}	Address
PD/PGM	Power down/Program
$\overline{CS}$	Chip Select
O_0-O_7	Outputs

MODE SELECTION

MODE \ PINS	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Deselect	Don't care	V_{IH}	+5	+5	High Z
Power Down	V_{IH}	Don't care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

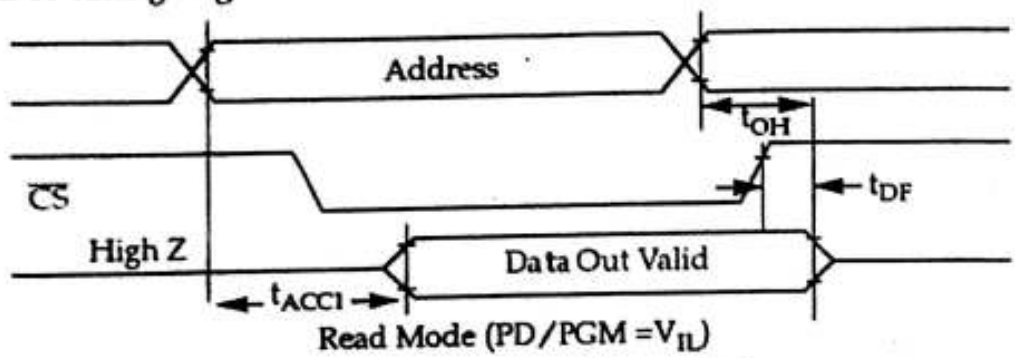
2 read

verify

Inhibit

EPROMs

2716 Timing diagram:



Read Mode (PD/PGM $=V_{IL}$)

Sample of the data sheet for the 2716 A.C. Characteristics.

Symbol	Parameter	Limits			Unit	Test Condition
		Min	Typ.	Max		
t_{ACC1}	Addr. to Output Delay		350	450	ns	PD/PGM= $\overline{CS}$ $=V_{IL}$
t_{OH}	Addr. to Output Hold	0			ns	PD/PGM= $\overline{CS}$ $=V_{IL}$
t_{DF}	Chip Deselect to Output Float	0		100	ns	PD/PGM=V_{IL}
...	...	...	...	...	...	...

This EPROM requires a wait state for use with the 8086 (*460ns* constraint).

Several EPROMs are available commercially, which are listed below in a tabular form. Out of these, the IC types 2708, 2716, 2732–although in use today, are increasingly being replaced by the newer varieties because of higher bit capacity.

ICs	No. of bits (in K)	Memory Organisation type
2708	8	1 K × 8
2716	16	2 K× 8
2732	32	4 K × 8
2764	64	8 K × 8
27128	128	16 K × 8
27256	256	32 K × 8
27512	512	64 K × 8

RAM types

SRAM (Static RAM)

- Storage cells are made of flip-flops and therefore they do not require refreshing to keep their data
- Cells handling one bit requires 6 or 4 transistors each, which is too many
- SRAMs are widely used for cache memory and battery-backed memory systems.
- Speeds as fast as 10ns. But limited in size ~256Kx8

- **DRAM (Dynamic RAM)**
  - Uses MOS capacitors to store a bit
  - Requires constant refreshing due to leakage (every 2ms – 4ms)
  - Advantages
    - High density (capacity) ~1GBX8
    - Cheaper cost per bit
    - Lower power consumption
  - Disadvantage
    - While it is being refreshed, data cannot be accessed
    - Larger access times
    - Too may pins due to large size

Comparison between Static and Dynamic RAM cells

S.No.	Static memory		Dynamic memory
1.	Stored data is retained as long as power remains ON.	1.	Stored data gets lost and repeated refreshing is required.
2.	Stored data do not change with time.	2.	Stored data changes with time.
3.	Consumes more power.	3.	Consumes less power than static memory.
4.	Expensive.	4.	Less expensive.
5.	These memories have less packing density.	5.	Higher packing density.
6.	These memories are not easy to construct.	6.	Simpler in construction.
7.	No refreshing required and easy in operation.	7.	Refreshing required with additional memory circuitry and hence complicated operation.
8.	No maintenance.	8.	Maintenance needed.

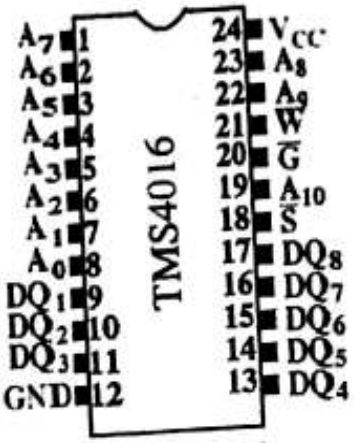
Static Ram (SRAM) Devices

Static RAM memory devices retain data for as long as DC power is applied. Because no special action (except power) is required to retain stored data, these devices are called *static memory*. They are also called *volatile memory* because they will not retain data without power. The main difference between a ROM and a RAM is that a RAM is written under normal operation, while a ROM is programmed outside the computer and is only normally read.

The SRAM stores temporary data and used when the size of the read/write memory is relatively small.

SRAMs

TI TMS 4016 SRAM (2K X 8):



2K x 8 SRAM

Pin(s)	Function
A_0-A_{10}	Address
DQ_0-DQ_7	Data In/Data Out
$\overline{S}$ ($\overline{CS}$)	Chip Select
$\overline{G}$ ($\overline{OE}$)	Read Enable
$\overline{W}$ ($\overline{WE}$)	Write Enable

Virtually identical to the EPROM with respect to the pinout.

However, access time is faster (250ns).

See the timing diagrams and data sheets in text.

SRAMs used for *caches* have access times as low as *10ns*.

SRAM	Density (bits)	Organization
4361	64K	64K× 1
4363	64K	16 K×4
4364	64K	8 K×8
43254	256K	64 K×4
43256A	256K	32 K×8
431000A	1M	128 K×8

DRAMs

DRAM:

SRAMs are limited in size (up to about 128K X 8).
DRAMs are available in much larger sizes, e.g., 64M X 1.

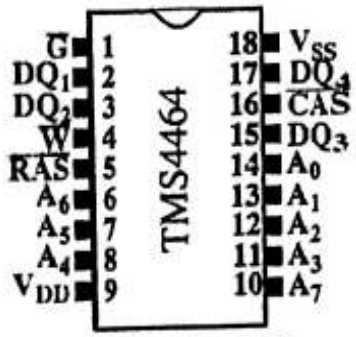
DRAMs MUST be refreshed (rewritten) every 2 to 4 ms
Since they store their value on an integrated capacitor that loses charge over time.

This refresh is performed by a special circuit in the DRAM which refreshes the entire memory using *256* reads.
Refresh also occurs on a normal read, write or during a special refresh cycle.

The large storage capacity of DRAMs make it impractical to add the required number of address pins.
Instead, the address pins are *multiplexed*.

DRAMs

TI TMS4464 DRAM (64K X 4):



64K x 4 DRAM

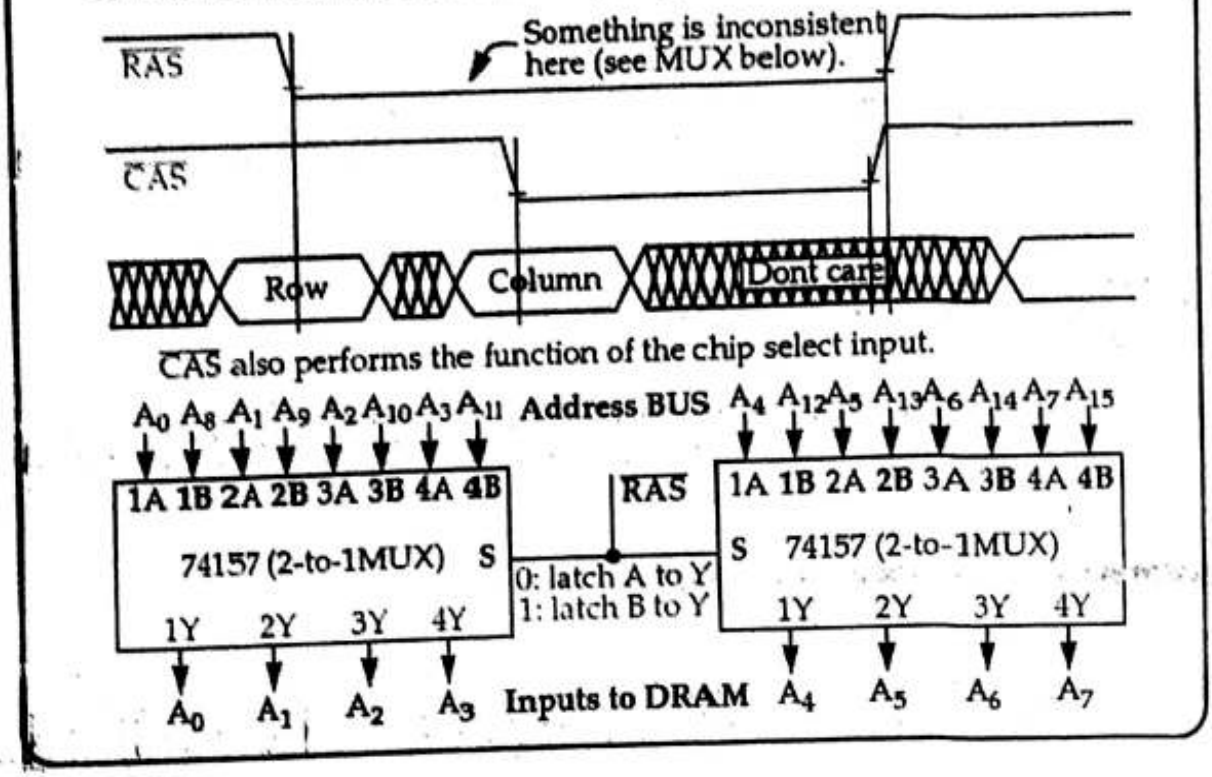
Pin(s)	Function
A_0-A_7	Address
DQ_0-DQ_4	Data In/Data Out
$\overline{RAS}$	Row Address Strobe
$\overline{CAS}$	Column Address Strobe
$\overline{G}$	Output Enable
$\overline{W}$	Write Enable

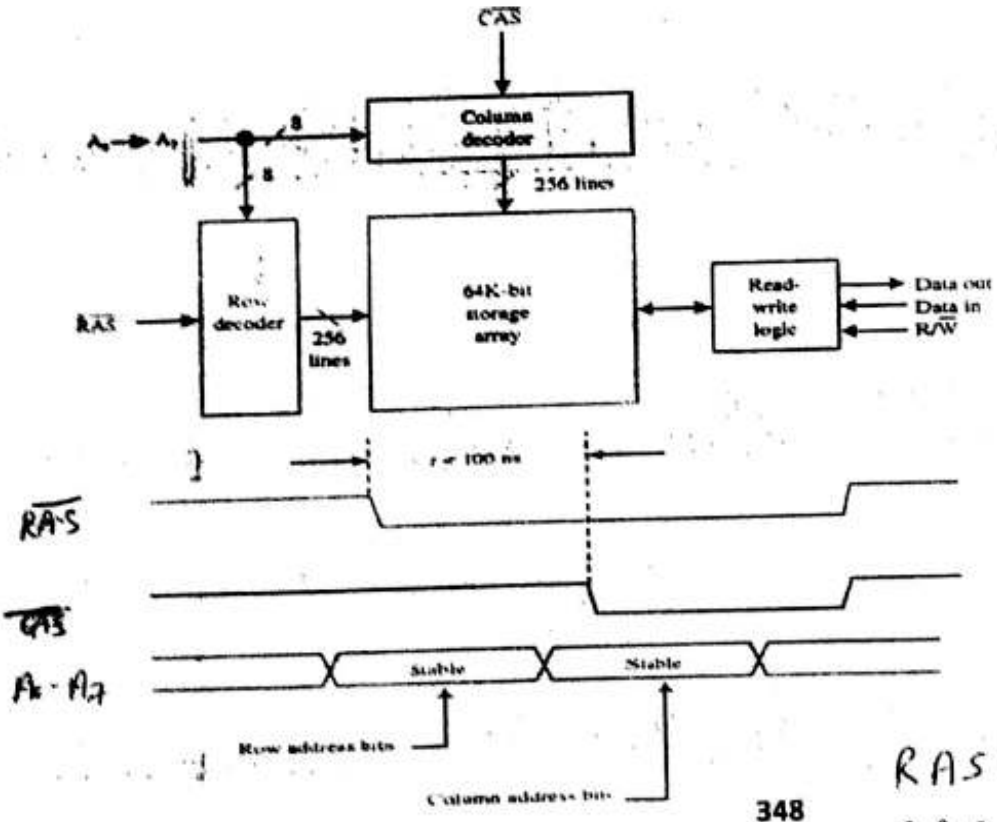
The TMS4464 can store a total of 256K bits of data.

It has 64K addressable locations which means it needs 16 address inputs, but it has only 8.
The row address (A_0 through A_7) are placed on the address pins and strobed into a set of internal latches.
The column addres (A_8 through A_{15}) is then strobed in using CAS.

DRAMs

TI TMS4464 DRAM (64K X 4) Timing Diagram:





- In DRAM, the 8 address lines are **latched accordingly** by the strobe of the RAS and CAS signals.
- For example: To load a 16 bit address into the DRAM 8 bits of the address are first latched by pulling RAS low, then other 8 bits are presented to A0-A7 and CAS is pulled low.

RAS: Row Address Strobe

CAS Column ~ ~

DRAMs

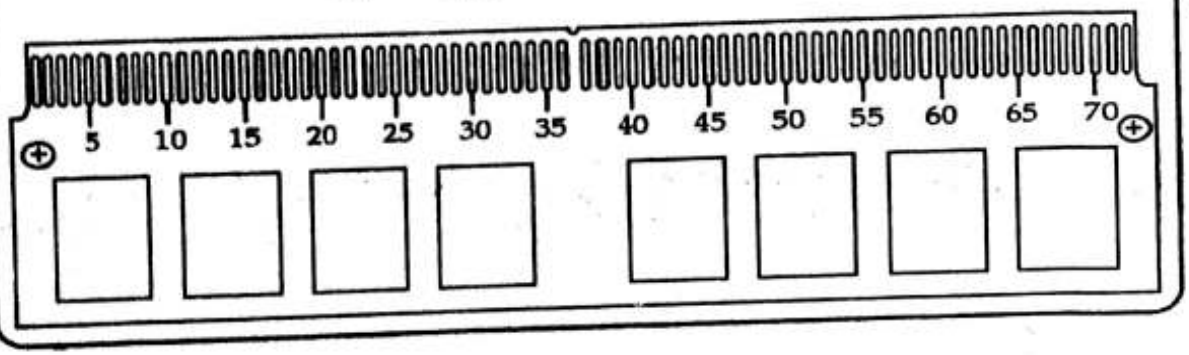
Larger DRAMs are available which are organized as *1M X 1, 4M X 1, 16M X 1, 64M X 1* (with *256M X 1* available soon).

DRAMs are typically placed on SIMM (Single In-line Memory Modules) boards.

30-pin SIMMs come in *1M X 8, 1M X 9* (parity), *4M X 8, 4M X 9.*

72-pin SIMMs come in *1/2/3/8/16M X 32* or *1M X 36* (parity).

V_{SS} $Addr_{0-11}$ $\overline{RAS}$ $\overline{W}$ NC
V_{CC} DQ_{0-31} $\overline{CAS}$ $\overline{PD}_{1-4}$



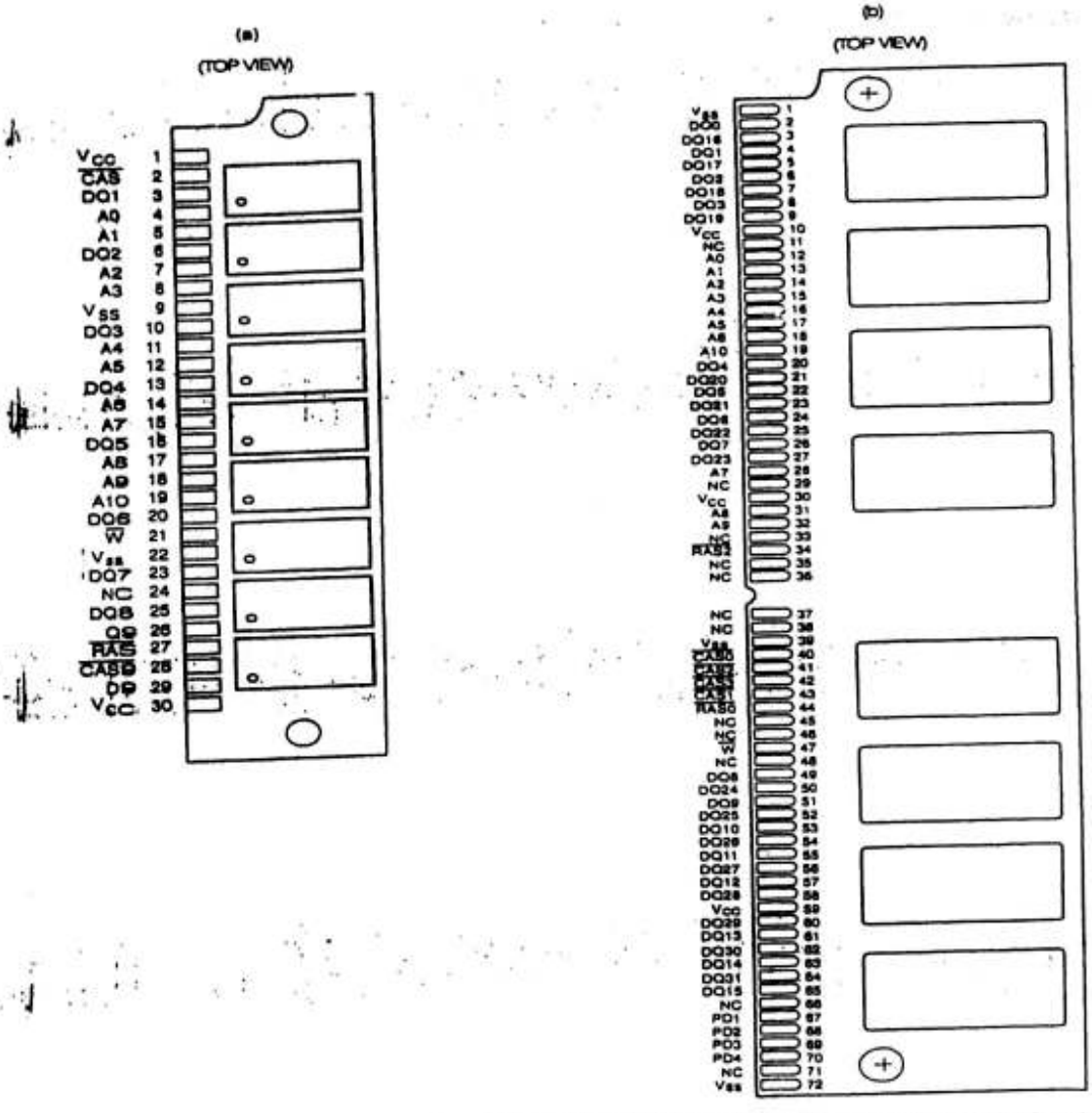
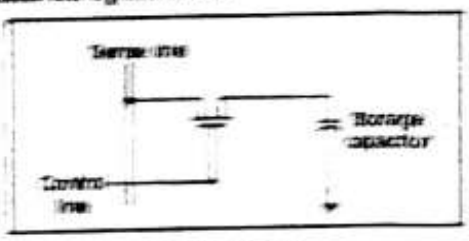


FIGURE 9–11 The pin-outs of the 30-pin and 72-pin SIMM. (a) A 30-pin SIMM organized as 4M × 9 and (b) a 72-pin SIMM organized as 4M × 36.

Draw a dynamic RAM (DRAM) cell and explain its operation.

Ans. A basic DRAM storage cell is shown below:



Dynamic RAM

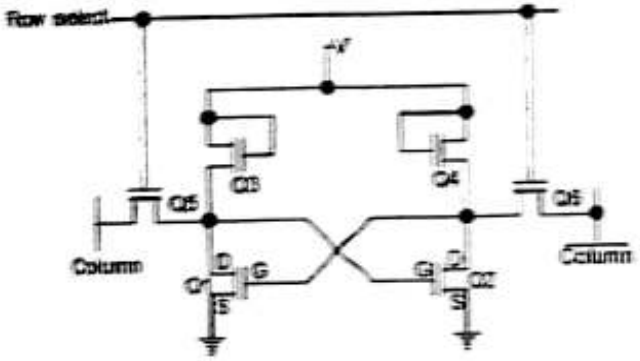
When column (sense) and row (control) lines go high, the MOSFET conducts and charges the capacitor. Again when the column and row lines go high, the MOSFET opens and the capacitor retains its charge. Thus, it can store a single bit. Since only a single MOSFET and a capacitor are employed to store a bit, the DRAM density is high. Here, the MOS transistor acts as a switch.

Some of the characteristics of a DRAM cell are

- Higher packing density.
- Charge leaks, thus refreshing needed.
- Extra hardware needed to implement refreshing operation.

Draw a static RAM (SRAM) cell and explain its operation.

Ans. A standard SRAM consists of six transistors connected to form a R-S flip-flop. A SRAM cell is shown below. The transistors Q_3 and Q_4 form the cross-coupling transistors required for latching.



Basic six-transistor static memory cell

Data is written into the cell by applying the data and its complement at the column and $\overline{\text{column}}$ inputs respectively, with Q_5 and Q_6 in the ON condition. Data can be read out from the column line after Q_5 and Q_6 are enabled.

Some of the characteristics of a SRAM cell are:

- It is volatile—i.e., data is lost on switch-off.
- When powered, the cell may assume either 1 or 0 state.
- Easy interfacing possible.
- No special timing circuits required.
- Bit density is low compared to DRAM.

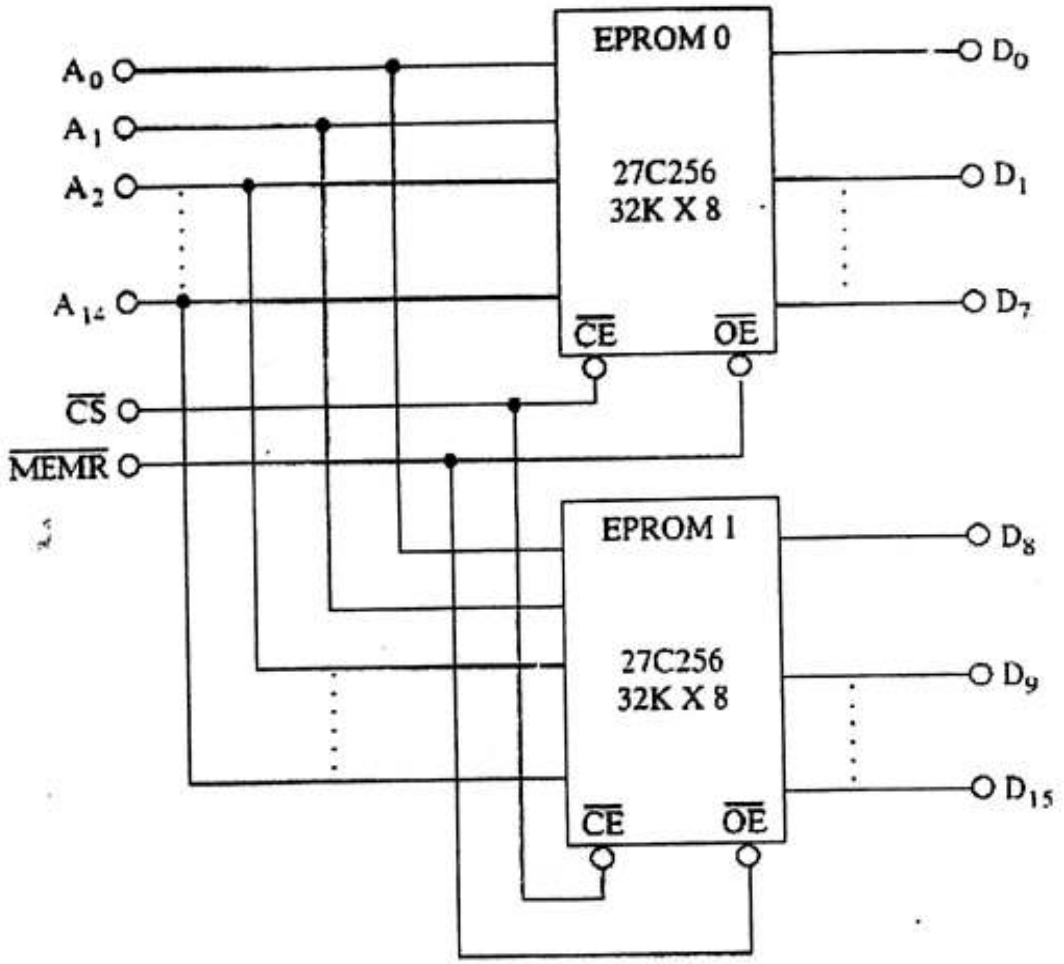
Memory expansion

In many applications, the microcomputer system requirement for memory is greater than what is available in a single device. There are two basic reasons for expanding memory capacity:

1. The byte-wide length is not large enough
2. The total storage capacity is not enough bytes.

Both of these expansion needs can be satisfied by interconnecting a number of ICs.

Example 1: show how to implement 32K× 16 EPROM using two 32K×8 EPROM?



Design a memory having size 16 × 8 from 16 × 4 memory modules.

A 16 × 8 memory module indicates that it can address 16 different memory addresses, each address location can store 1 byte of data/instruction. The design is given below:

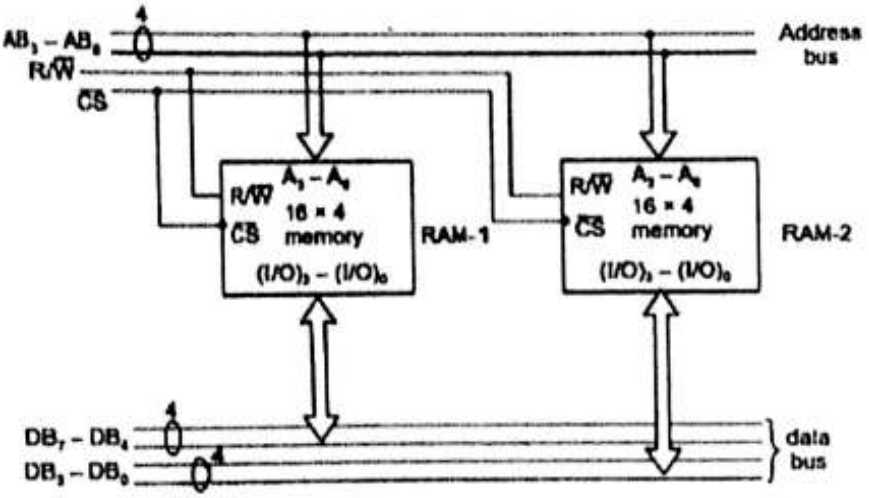


Diagram showing a 16 × 8 memory obtained from two 16 × 4 memories

The address lines $AB_3 - AB_0$ can address 16 different addresses (from 0000 to 1111) and connected to the two RAMs as shown. Chip selection is on the basis of $\overline{CS}$ signal while R/W signal governs of whether reading (from memory) or writing (into the memory) is to be done.

Once a particular address has been selected (by $AB_3 - AB_0$ lines), $DB_7 - DB_4$ stores the upper nibble of data in the left RAM (RAM-1) while $DB_3 - DB_0$ stores the lower nibble of the data in the right RAM (RAM-2).

Develop a 32 × 4 memory module by combining two 16 × 4 memory chips.

The interconnections between the two 16 × 4 memory chips is shown below which yields a 32 × 4 memory module.

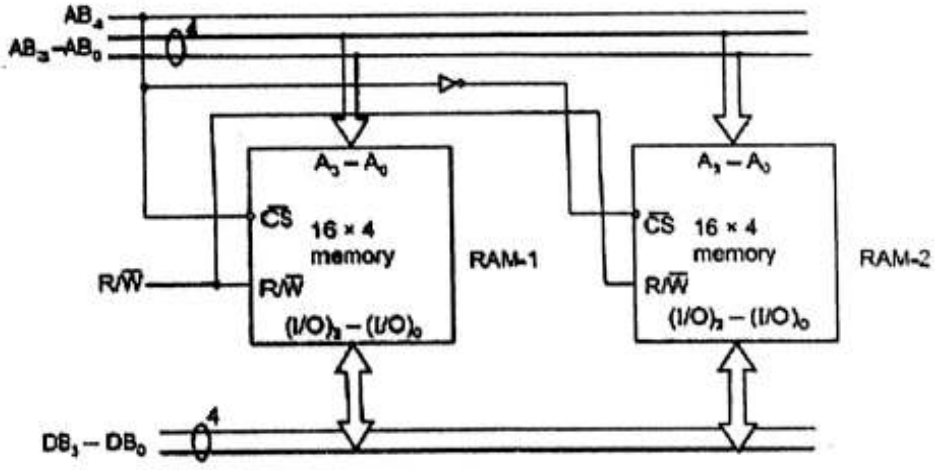
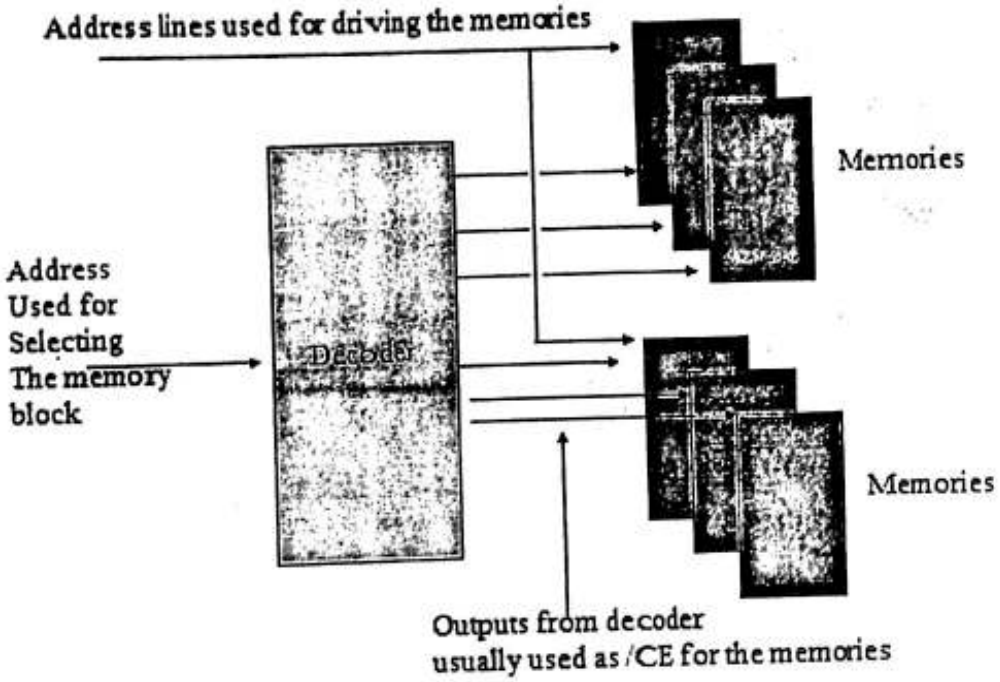


Diagram showing a 32 × 4 memory obtained from two 16 × 4 memory modules

ADDRESS DECODING

In order to attach a memory device to the microprocessor, it is necessary to decode the address from the microprocessor to make the memory function at a unique section or partition of the memory map. Without an address decoder, only one memory device can be connected to a microprocessor, which would make it virtually useless. In this section, we describe a few of the more common address-decoding techniques, as well as the decoders that are found in many systems.



Memory Address Decoding

The processor can usually address a memory space that is *much larger* than the memory space covered by an individual memory chip.

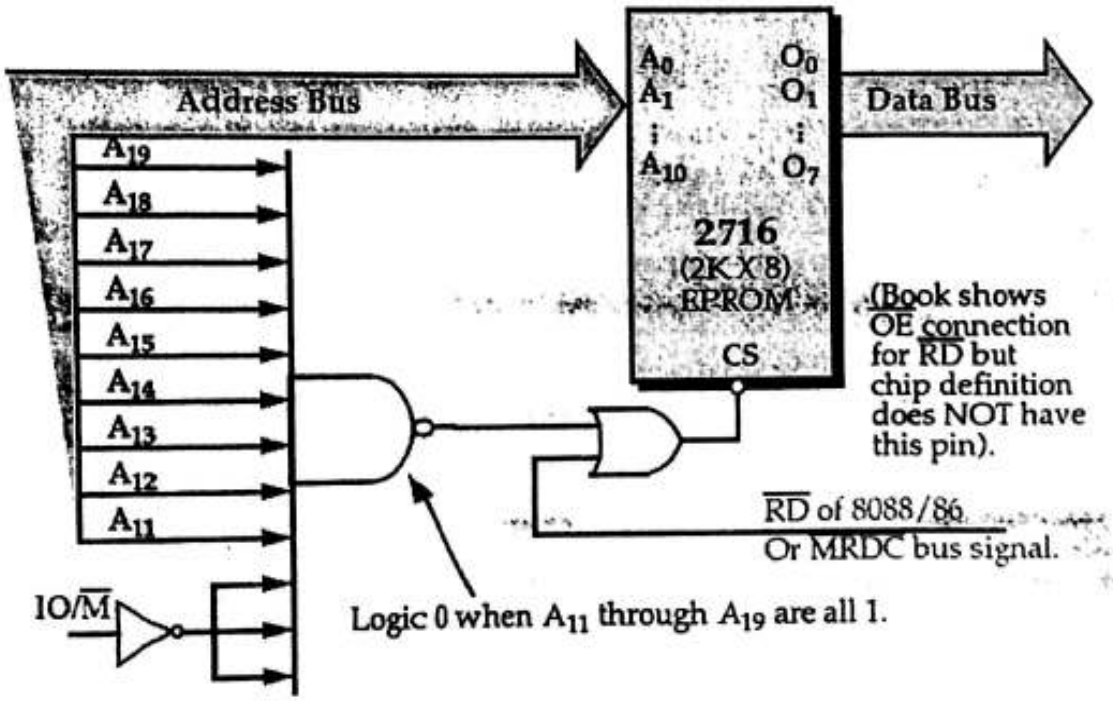
In order to splice a memory device into the address space of the processor, decoding is necessary.

For example, the 8088 issues *20-bit* addresses for a total of 1MB of memory address space.

However, the BIOS on a 2716 EPROM has only 2KB of memory and *11* address pins.

A decoder can be used to decode the additional 9 address pins and allow the EPROM to be placed in any 2KB section of the 1MB address space.

Memory Address Decoding



Memory Address Decoding

To determine the address range that a device is mapped into:

A_{19} - A_{11} : A_{10} - A_0

```
1111 1111 1XXX XXXX XXXX
          ↓
1111 1111 1000 0000 0000 (FF800H)
          ↓ To
1111 1111 1111 1111 1111 (FFFFFH)
```

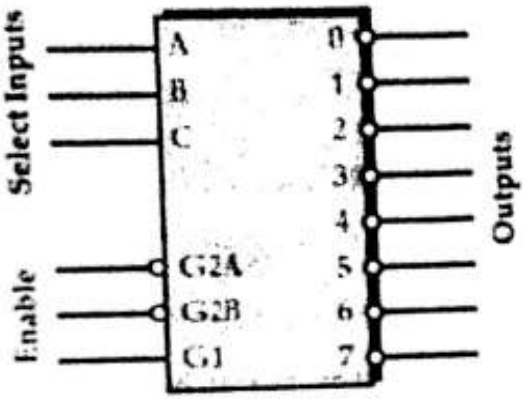
This 2KB memory segment maps into the **reset location** of the 8086/8088 (FFFF0H).

NAND gate decoders are not often used.

Rather the 3-to-8 Line Decoder (74LS138) is more common.

Memory Address Decoding

The 3-to-8 Line Decoder (74LS138)

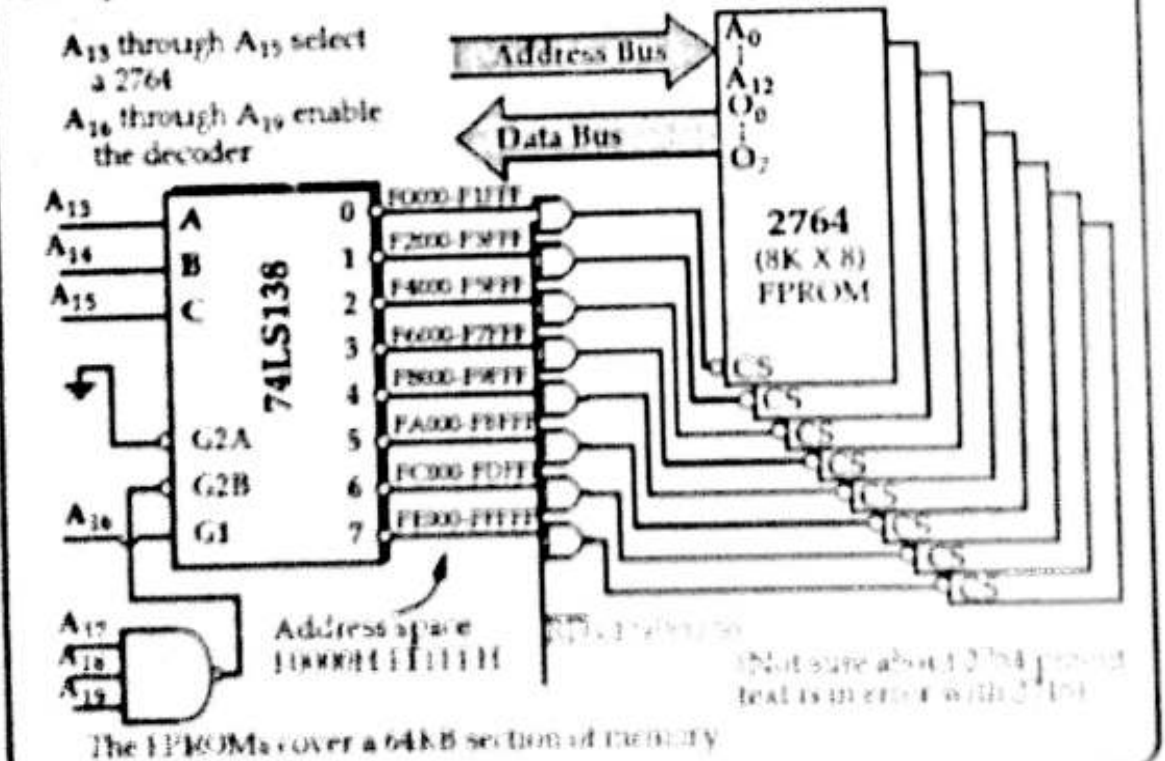


Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

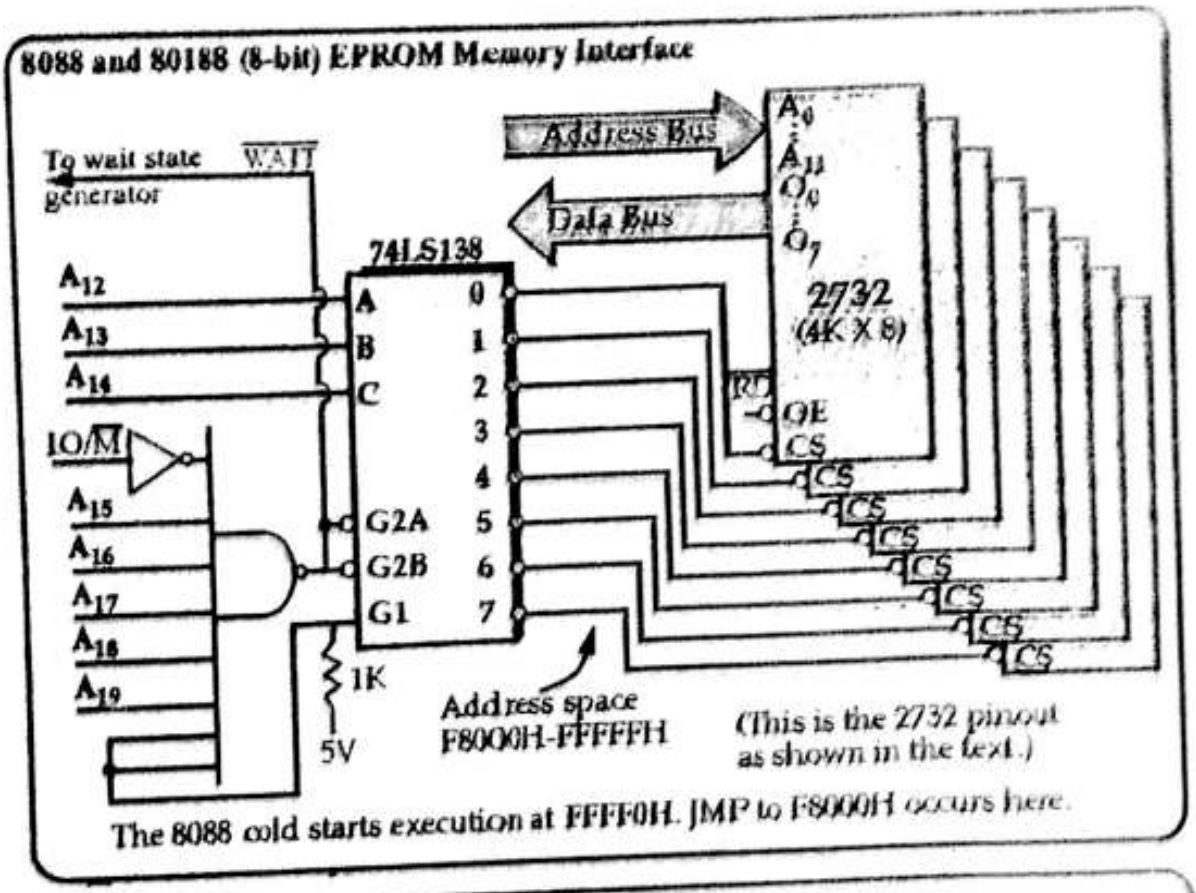
Note that all *three* Enables (G2A, G2B, and G1) must be active, e.g. low, low and high, respectively

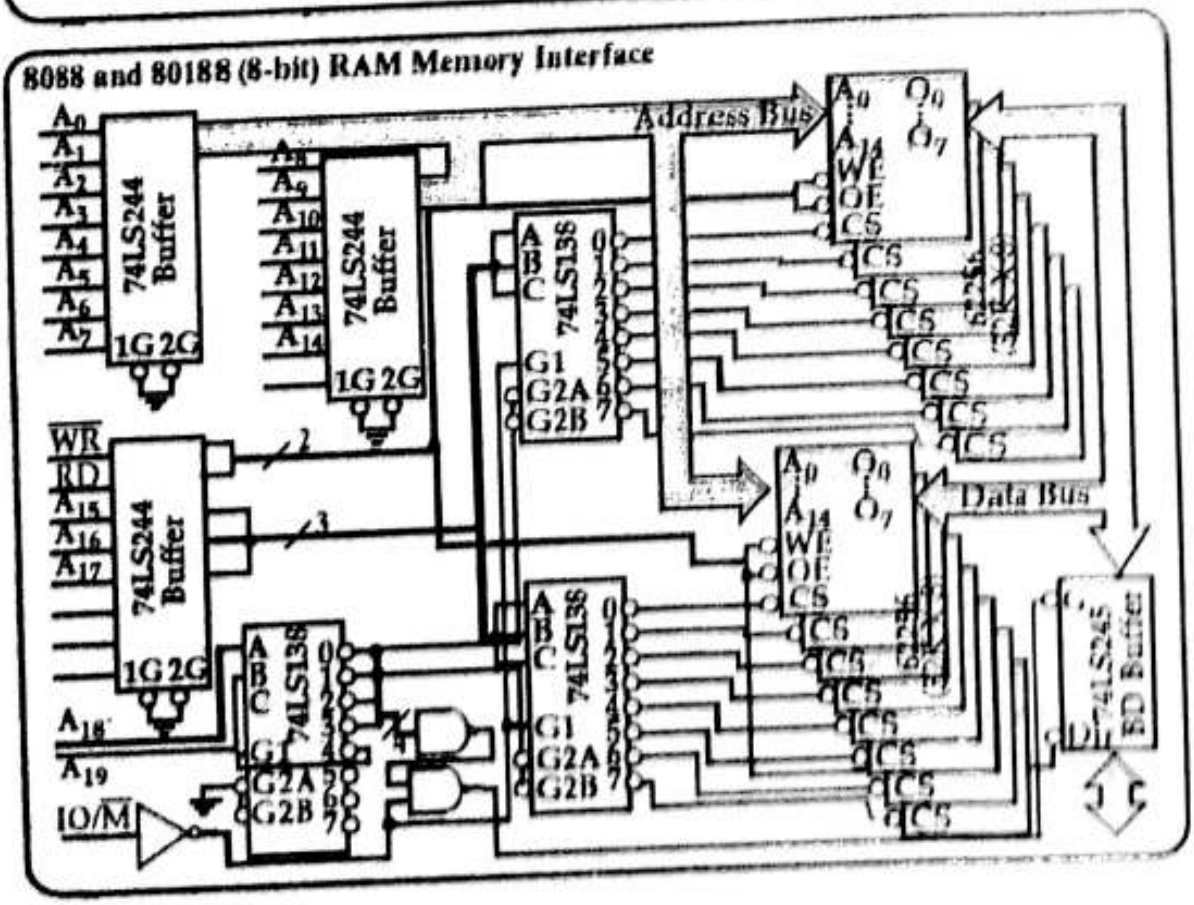
Each output of the decoder can be attached to an 2764 EPROM (*8K X 8*)

Memory Address Decoding



The EPROMs cover a 64kB section of memory


8088 and 80188 (8-bit) EPROM Memory Interface
To wait state generator
WAIT
Address Bus
Data Bus
74LS138
2732
(4K X 8)
IO/M
1K
5V
Address space
F8000H-FFFFFH
(This is the 2732 pinout as shown in the text.)
The 8088 cold starts execution at FFFF0H. JMP to F8000H occurs here.

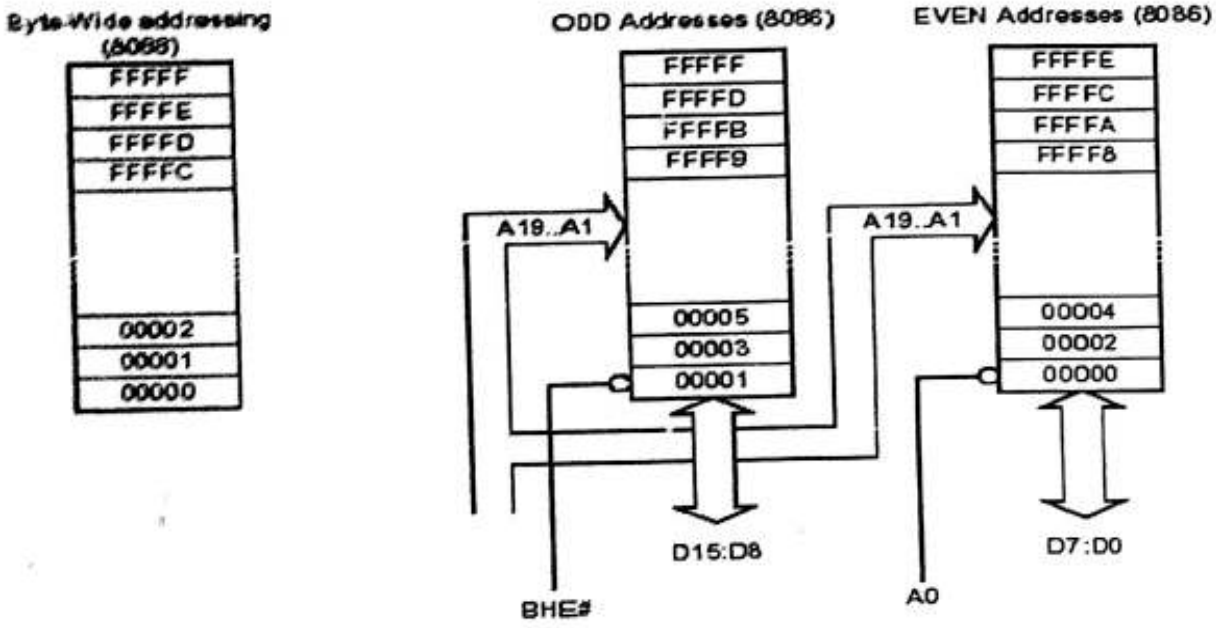

8088 and 80188 (8-bit) RAM Memory Interface
Address Bus
Data Bus
74LS244 Buffer
74LS138
74LS245 BD Buffer
IO/M

8086 Memory Interface

The total address space 1 MB of 8086 is divided into 2 banks of memory—each bank of maximum size 512 KB. One is called the high order memory bank (or high bank) and the other low order memory bank (or low bank). Low bank, high bank or both banks can be accessed by utilizing two signals BHE and A0.The following shows the three possible references to memory.

The low and high order memory banks correspond to even and odd banks respectively.

The $\overline{CS}$ signal of low order memory bank is selected when $\overline{CS}$ = 0. Since A_0 (lowest address bus line) is connected to $\overline{CS}$, hence A_0 must have be to low for the low order bank to be selected. That is why the low order bank corresponds to even bank. Similarly the high order bank is selected when A_0 = 1. Hence, the higher order bank is called odd bank.

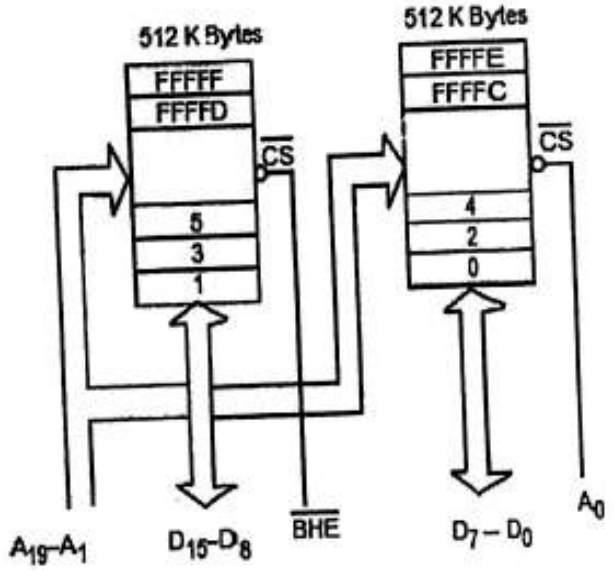


$\overline{BHE}$	$\overline{BLE}$ (A0)	Function	Example
0	0	Both banks enabled (16 bit)	MOV [1000H],AX
0	1	High bank enabled (8 bit)	MOV [1001H],AL
1	0	Low bank enabled (8 bit)	MOV [1000H],AL
1	1	No banks enabled	----------

$\overline{BHE}$	A_0	Processing
0	0	Both Banks Active 16-bit world transfer on $AD_{15} \Leftrightarrow AD_0$
0	1	Only High bank Active (One byte from/to odd address on $AD_{15} \Leftrightarrow AD_8$)
1	0	Only Low bank Active (One byte from/to even address on $AD_7 \Leftrightarrow AD_0$)
1	1	No Bank Active

The high bank is selected for A_0=1 and $\overline{BHE}$=0 and is connected to D_{15}–D_8 while the low bank is selected for A_0=0 and $\overline{BHE}$=1. Neither low bank nor high bank would selected for A_0=1 and $\overline{BHE}$=1.

Fig. shows how the total address space (1MB) of 8086 is physically implemented by segregating it into low and high banks. It also shows that $\overline{CS}$ signal of the high bank is connected to $\overline{BHE}$ while the $\overline{CS}$ signal of the low bank is connected to A_0.



Selection of high and low banks of 8086

Draw the diagrams of (a) even-addressed byte transfer (b) odd-addressed byte transfer (c) even-addressed word transfer and (d) odd-addressed word transfer.

A, B are representing the addresses while (A), (B) represent the content of address locations A and B respectively.

Figures (a) and (b) correspond to byte transfers for even and odd-addressed memory locations respectively. The shaded memory location indicates that the content of that particular memory location comes out either via higher byte data bus (D_{15}–D_8) or lower byte data bus (D_7–D_0) respectively.

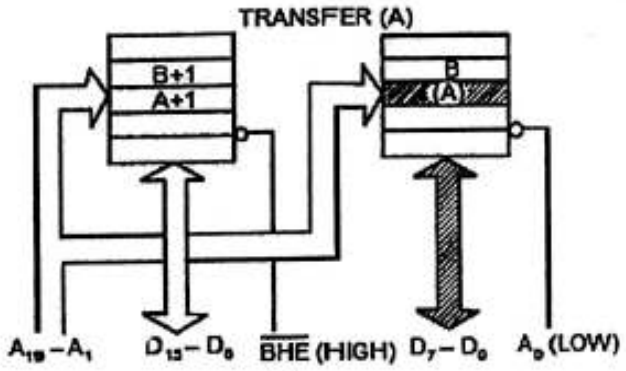
Figures (a), (b) and (c) complete the data transfer in one bus cycle only.

For Figure (a), $\overline{BHE} = 1, A_0 = 0$

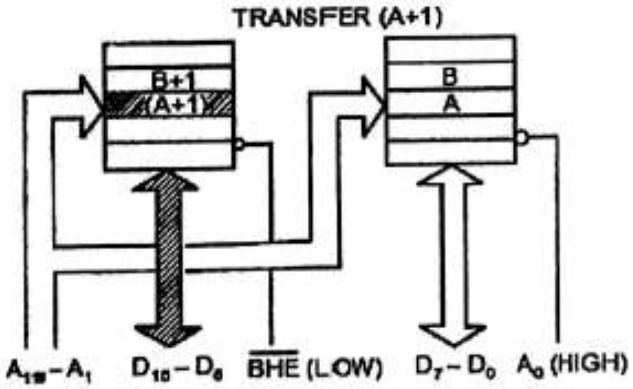
For Figure (b), $\overline{BHE} = 0, A_0 = 1$

For Figure (c), $\overline{BHE} = 0, A_0 = 0$

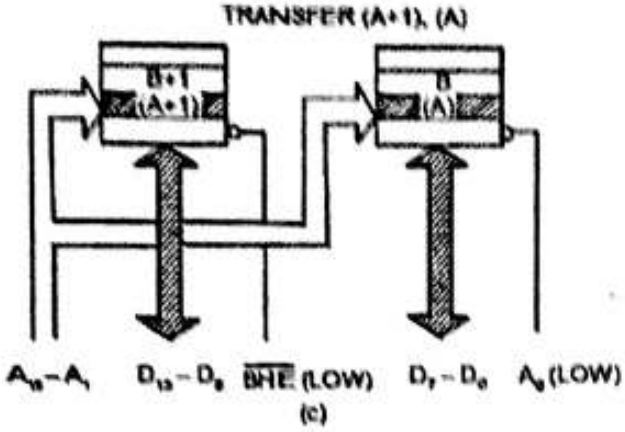
Figure (d) corresponds to an odd-addressed word transfer and it takes two bus cycles to complete this transfer.

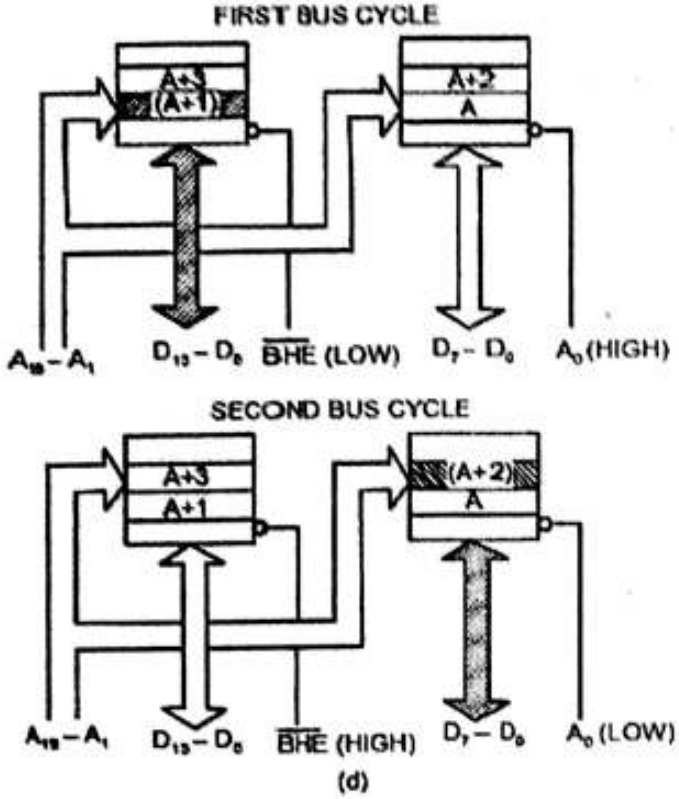


(a)



(b)

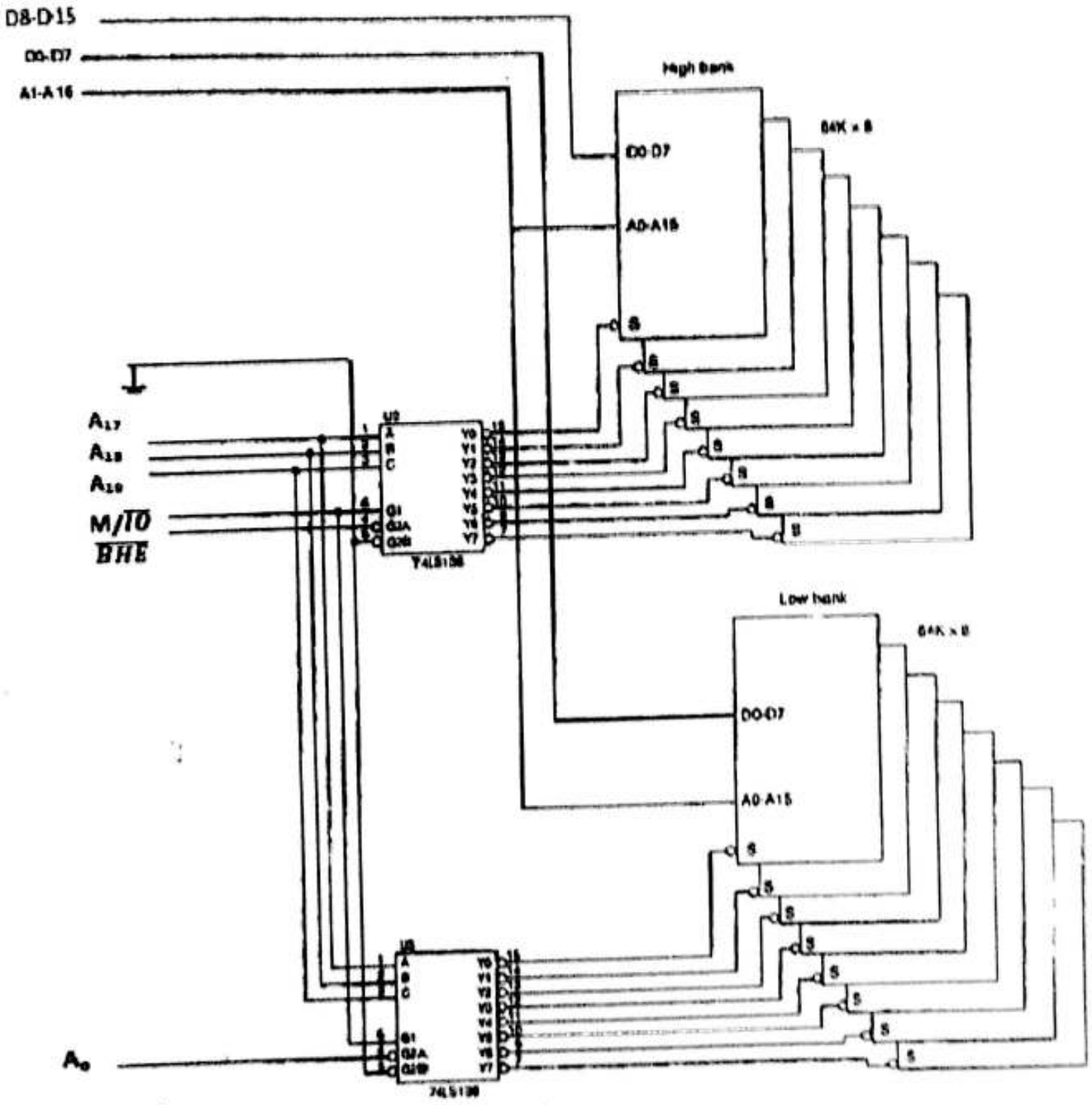




(a) Even-addressed byte transfer by the 8086.
(b) Odd-addressed byte transfer by the 8086.
(c) Even-addressed word transfer by the 8086.
(d) Odd-addressed word transfer by the 8086.

Example

Design a 8086 memory system consisting of 1Mbytes, Using 64K × 8 memory



Example

Design an 8KX8 RAM module using 2KX8 RAM chips. The module should be connected on an 8-bit processor with a 16-bit address bus, and occupy the address range starting from the address A000. Show the circuit and the memory map?

- Number of memory devices needed = 8K/2K = 4
- Decoder needed = 2X4
- Number of address lines on each 2KX8 memory chip = 11

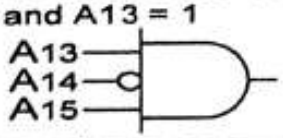
  $2^m = 2K = 2^1 \times 2^{10} = 2^{11} \Rightarrow$ (A0..A10)
- Decoder needed = 2X4

  $\Rightarrow$ 2 address lines are needed for the decoder. $\Rightarrow$ (A11..A12)
- Number of address lines needed for the address selection circuit

  = 16 - 11 - 2 = 3 $\Rightarrow$ (A13, A14 A15)

Starting Address = A000 = 1010-0000-0000-0000
==> A15 = 1, A14 = 0 and A13 = 1

Address Selection Circuit



A15	A14	A13	A12	A11	A10 ..... A0	Mem. Map	
0	0	0	0	0	0 0	0000	Not Used
1	0	0	1	1	1 1	9FFF	
1	0	1	0	0	0 0	A000	RAM1
1	0	1	0	0	1 1	A7FF	
1	0	1	0	1	0 0	A800	RAM2
1	0	1	0	1	1 1	AFFF	
1	0	1	1	0	0 0	B000	RAM3
1	0	1	1	0	1 1	B7FF	
1	0	1	1	1	0 0	B800	RAM4
1	0	1	1	1	1 1	BFFF	
1	1	0	0	0	0 0	C000	Not Used
1	1	1	1	1	1 1	FFFF	

Solution:

62256 SRAM chips:

⇨ 256/8 =32 ⇨ 32KX8

Number of chips needed:

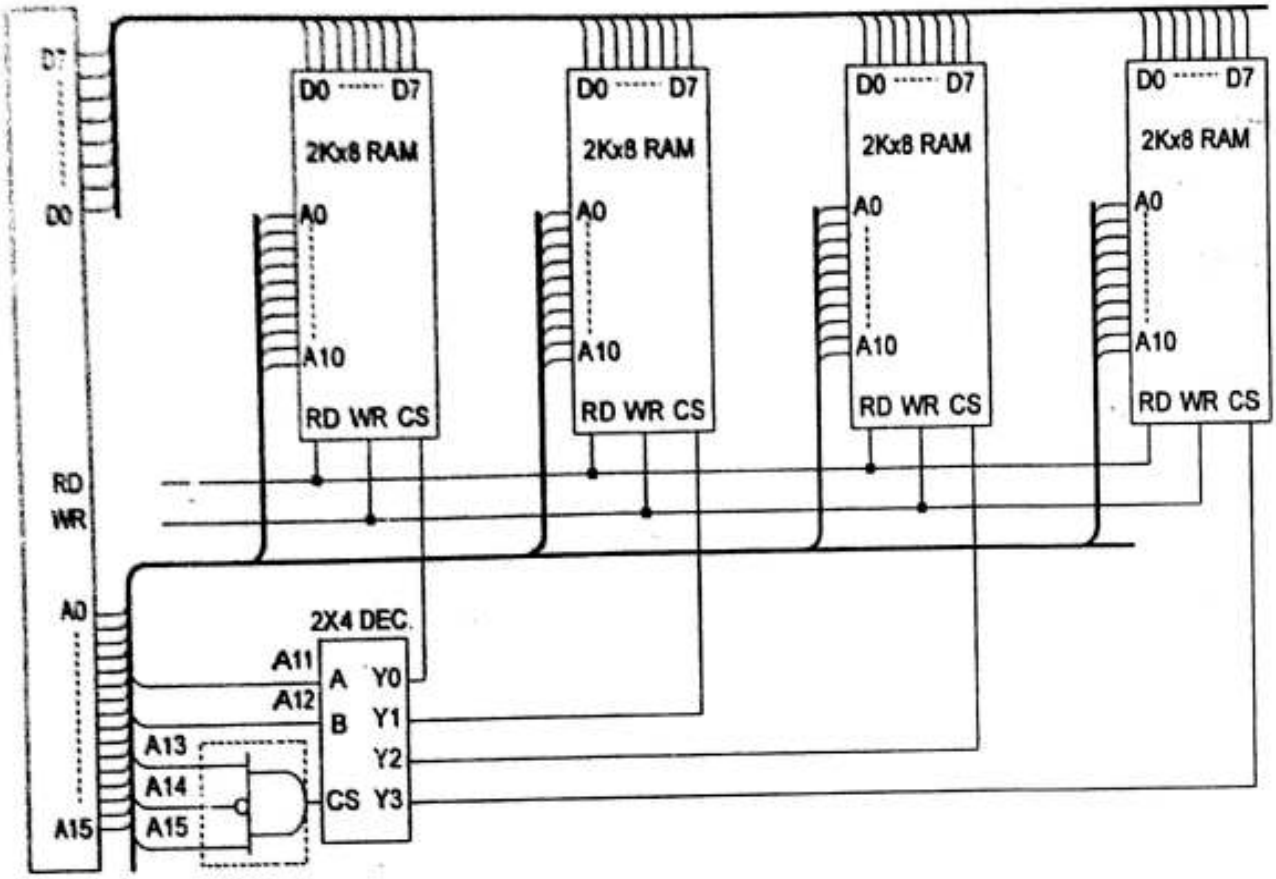
⇨ 128K/32K = 4

Number of address lines:

⇨ $32K = 2^5K = 2^5 * 2^{10} = 2^{15}$

⇨ 15 address lines (A0 .. A14)

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13} •••	A_0	Memory Map
1	1	0	0	0	0	0 •••	0	C0000 RAM1
1	1	0	0	0	1	1 •••	1	C7FFF
1	1	0	0	1	0	0 •••	0	C8000 RAM2
1	1	0	0	1	1	1 •••	1	CFFFF
1	1	0	1	0	0	0 •••	0	D0000 RAM3
1	1	0	1	0	1	1 •••	1	D7FFF
1	1	0	1	1	0	0 •••	0	D8000 RAM4
1	1	0	1	1	1	1 •••	1	DFFFF



Example

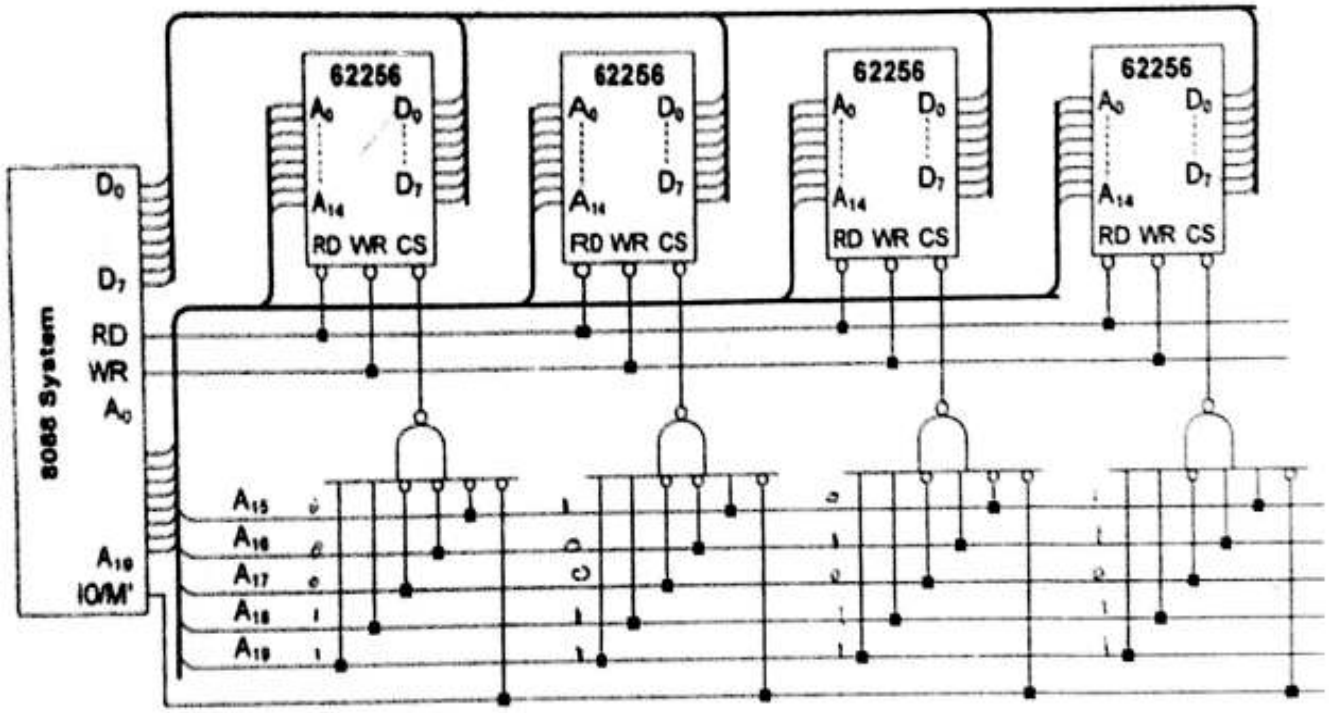
Show how a 128Kbyte RAM module can be connected on an 8088 system using 62256 SRAM chips, occupying the address range starting from the address C0000H. Use the following address decoding circuits:

1- NAND Gates.

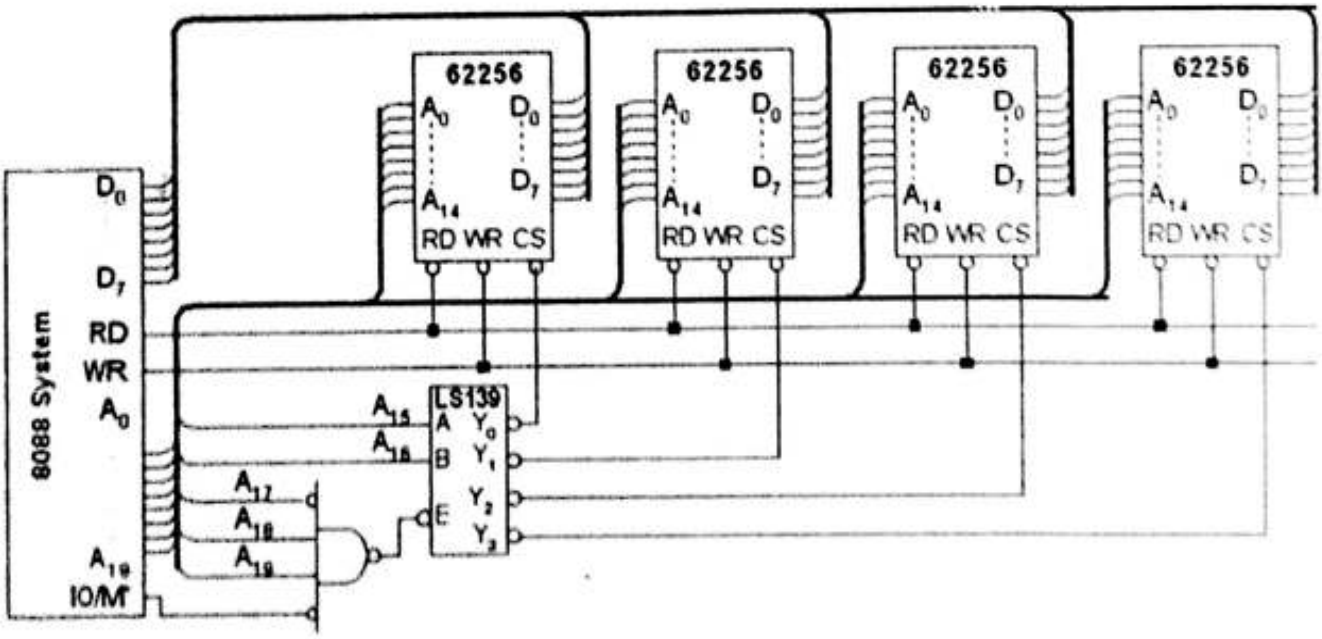
2- DECODER and NAND gate.

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1- NAND Gates

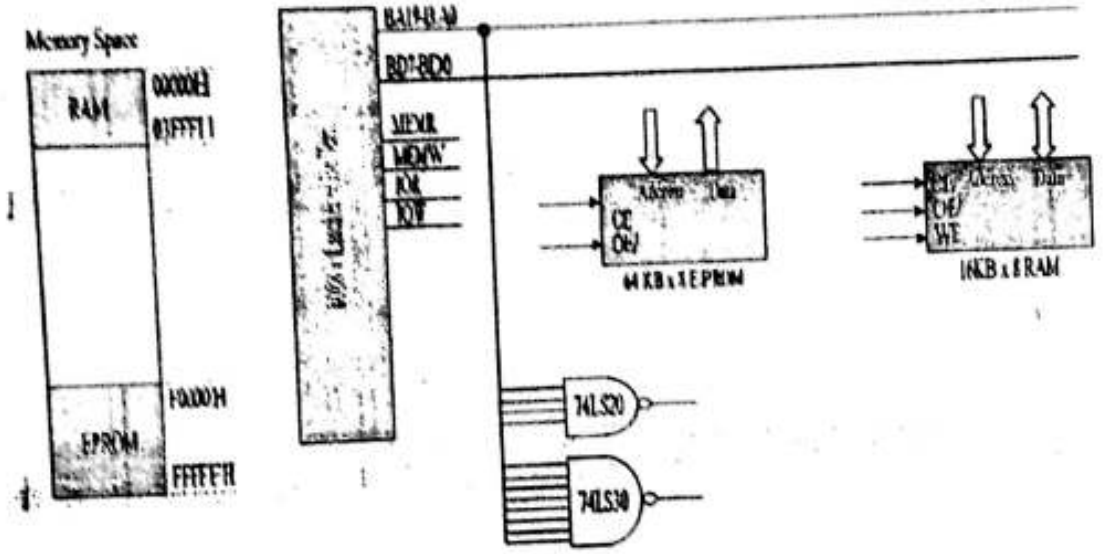


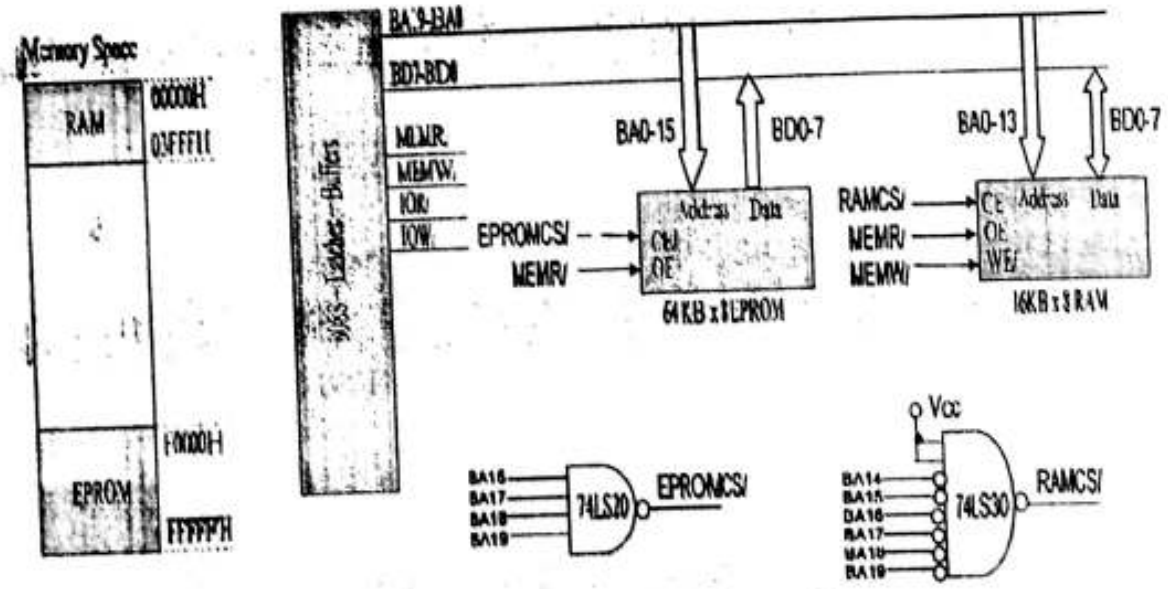
2- Decoder and NAND Gate



Example

Interface a 16KB RAM and a 64KB EPROM to an 8088 buffered system. Assume the starting address of the RAM is 0H and the starting address of the EPROM is F000OH. Use NAND gates, as needed, to generate the chip selects.





29

Example:

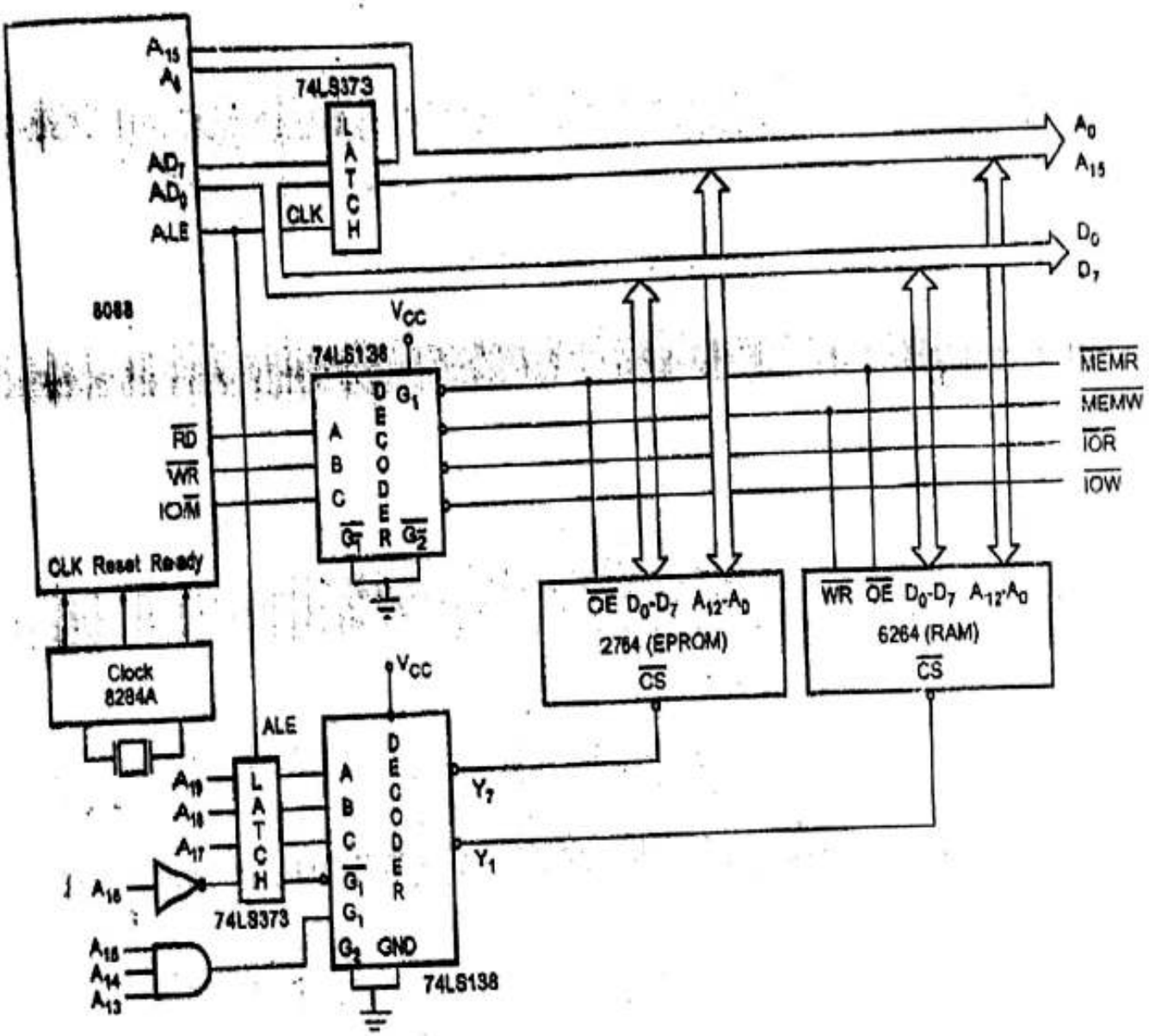
- Design an 8086 based system with the following specifications:
  - 8086 in minimum mode
  - 64 KB EPROM
  - 64 KB RAM
- Draw the complete schematic of the design indicating address map.
- Sol: 8086 is 16 bit µp so it is necessary to have odd and even memory banks.
- Two 32 KB EPROMs and two 32 KB RAMs1
- For 32 KB RAM & EPROM need 15 address lines (A1-A15)
- A0 & BHE are used to select even and odd banks

Memory Map

Memory Map :

$\overline{BHE}$	A_{19} A_{18} A_{17} A_{16}	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	Address	Memory
1	1 1 1 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	F0000H	Even
1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 0	FFFFFH	EPROM1
0	0 1 1 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	F0001H	Odd
0	0 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	FFFFHH	EPROM2
1	1 0 0 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	30000H	Even
1	1 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 0	3FFFEH	RAM1
0	0 0 0 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	30001H	Odd
0	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	3FFFFH	RAM2

Interfacing 64 K RAM & 64 K EPROM



Interfacing 8 K RAM and 8 K EPROM with 8088 in minimum mode

31

12

Memory Organisation

1. Mention the address capability of 8086 and also show its memory map.

Ans. 8086, via its 20-bit address bus, can address 2^{20} = 1,048,576 or 1 MB of different memory locations. Thus the memory space of 8086 can be thought of as consisting of 1,048,576 bytes or 524,288 words.

The memory map of 8086 is shown in Fig. 12.1, where the whole memory space starting from 00000 H to FFFFF H is divided into 16 blocks—each one consisting of 64 KB. This division is arbitrary but at the same time a convenient one—because the most significant hex digit increases by 1 with each additional block. Thus, 30000 H memory location is 65,536 bytes higher in memory than the memory location 20000 H.

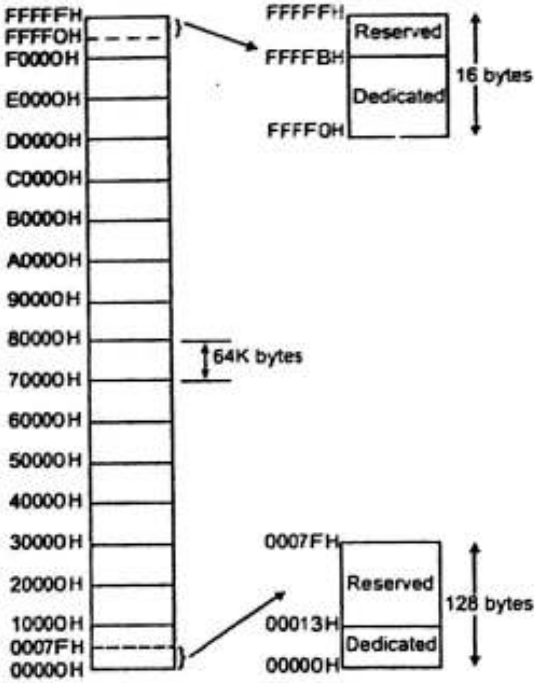


Fig. 12.1: Memory map for the 8086 microprocessor. Some memory locations are dedicated or reserved.

The lower and upper ends of the memory map are shown separately—earmarking some spaces as reserved and some as 'dedicated'.

The reserved locations are meant for future hardware and software needs while the dedicated locations are used for processing of specific system interrupts and reset functions.

2. Mention the different types of memory segmentations of 8086.

Ans. The different memory segmentations done in case of 8086 are

- Continuous
- partially overlapped
- fully overlapped and
- disjointed

This is shown in Fig.12.2.

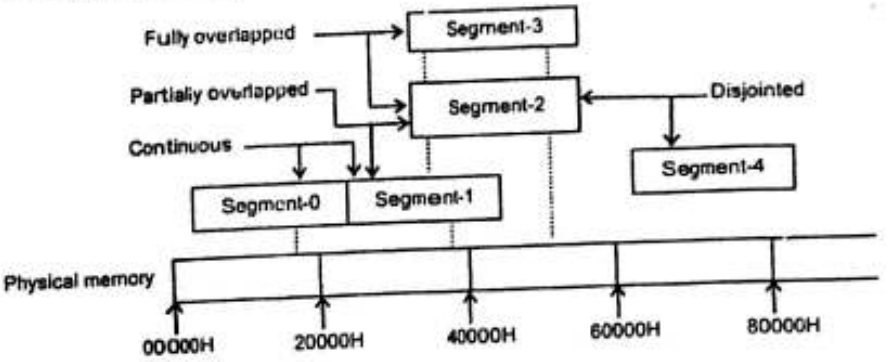


Fig. 12.2: Depiction of different types of segments

In the figure,

Segments-0	and	1 ⟶ Continuous
Segments-1	and	2 ⟶ Partially overlapped
Segments-2	and	3 ⟶ Fully overlapped
and Segments-2	and	4 ⟶ Disjointed

3. Describe memory segmentation scheme of 8086. What is meant by currently active segments?

Ans. 1 MB memory of 8086 is partitioned into 16 segments—each segment is of 64 KB length. Out of these 16 segments, only 4 segments can be active at any given instant of time—these are code segment, stack segment, data segment and extra segment. The four memory segments that the CPU works with at any time are called currently active segments. Corresponding to these four segments, the registers used are Code Segment Register (CS), Data Segment Register (DS), Stack Segment Register (SS) and Extra Segment Register (ES) respectively.

Each of these four registers is 16-bits wide and user accessible—i.e., their contents can be changed by software

The code segment contains the instruction codes of a program, while data, variables and constants are held in data segment. The stack segment is used to store interrupt and subroutine return addresses.

The extra segment contains the destination of data for certain string instructions. Thus 64 KB are available for program storage (in CS) as well as for stack (in SS) while 128 KB of space can be utilised for data storage (in DS and ES).

One restriction on the base address (starting address) of a segment is that it must reside on a 16-byte address memory—examples being 00000 H, 00010 H or 00020 H, etc.

4. Mention the maximum size of memory that can be active for 8086.

Ans. The maximum size of active memory for 8086 is 256 KB. The break-up being
64 KB for program
64 KB for stack and
128 MB for data.

5. Why memory segmentation is done for 8086?

Ans. Memory segmentation, as implemented for 8086, gives rise to the following advantages:

- Although the address bus is 20-bits in width, memory segmentation allows one to work with registers having width 16-bits only.
- It allows instruction code, data, stack and portion of program to be more than 64 KB long by using more than one code, data, extra segment and stack segment.
- In a time-shared multitasking environment when the program moves over from one user's program to another, the CPU will simply have to reload the four segment registers with the segment starting addresses assigned to the current user's program.
- User's program (code) and data can be stored separately.
- Because the logical address range is from 0000 H to FFFF H, the same can be loaded at any place in the memory.

6. Discuss logical address, base segment address and physical address.

Ans. The logical address, also goes by the name of effective address or offset address (also known as offset), is contained in the 16-bit IP, BP, SP, BX, SI or DI.

The 16-bit content of one of the four segment registers (CS, DS, ES, SS) is known as the base segment address.

Offset and base segment addresses are combined to form a 20-bit physical address (also called real address) that is used to access the memory. This 20-bit physical address is put on the address bus ($AD_{19} - AD_0$) by the BIU.

7. Describe how the 20-bit physical address is generated.

Ans. The 20-bit physical (real) address is generated by combining the offset (residing in IP, BP, SP, BX, SI or DI) and the content of one of the segment registers CS, DS, ES or SS. The process of combination is as follows:

The content of the segment register is internally appended with 0 H (0000 H) on its right most end to form a 20 bit memory address—this 20-bit address points to the start of the segment. The offset is then added to the above to get the physical address.

Fig. 12.3 shows pictorially the actual process of generating a 20-bit physical address.

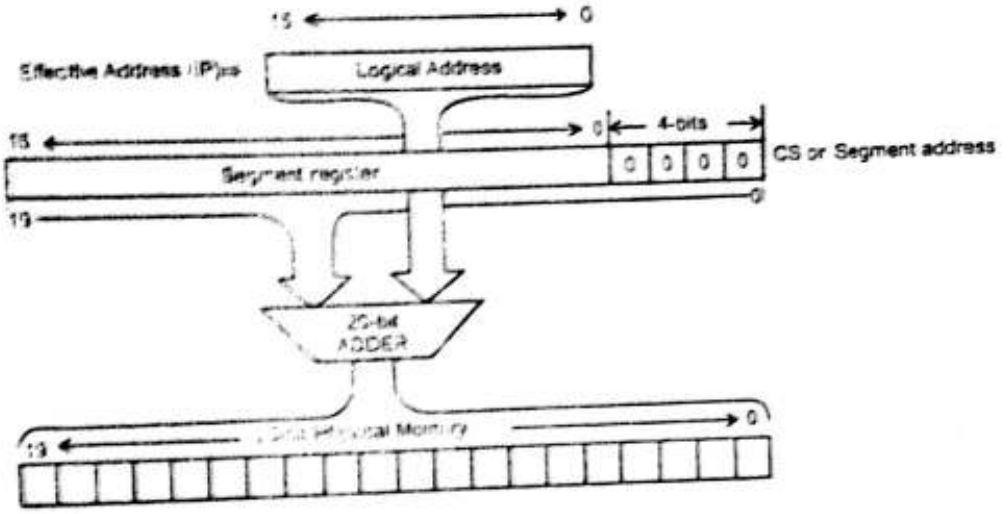


Fig. 12.3: Physical address generation

Thus, Physical Address = Segment Register content 16 D + Offset.

8. Although 8086 is a 16-bit μP, it deals with 8-bit memory. Why?

Ans. This is so for the following two reasons:

- It enables the microprocessor to work with both on bytes and words. This is very important because many I/O devices such as printers, terminals, modems etc, transfer ASCII coded data (7 or 8 bits).
- Quite a few of the operation codes of 8086 are single bytes while so many other instructions are there which vary from 2 to 7 bytes. By working with byte-width memory, these varied opcodes can easily be handled.

9. Is the flat scheme of memory applied for 8086 μP?

Ans. No, the flat (or unsegmented) scheme of memory is not applied for 8086 μP, because the memory of the same is a segmented one. In flat scheme, the entire memory space is thought of as a single addressable memory unit.

The flat scheme can be applied for 8086 by initialising all the segment registers with identical (or same) base address. Then all memory operations will refer to the same memory space.

10. Describe how memory is organised for 8086 μP?

Ans. The total address space 1 MB of 8086 is divided into 2 banks of memory—each bank of maximum size 512 KB. One is called the high order memory bank (or high bank) and the other low order memory bank (or low bank).

Low bank, high bank or both banks can be accessed by utilizing two signals $\overline{BHE}$ and A_0. Table 12.1 shows the three possible references to memory.

Table 12.1: Memory references

$\overline{BHE}$	A_0	Processing
0	0	Both Banks Active 16-bit word transfer on $AD_{15} \leftrightarrow AD_8$
0	1	Only High bank Active (One byte from/to odd address on $AD_{15} \leftrightarrow AD_8$)
1	0	Only Low bank Active (One byte from/to even address on $AD_7 \leftrightarrow AD_0$)
1	1	No Bank Active

The high bank is selected for A_0=1 and $\overline{BHE}$=0 and is connected to D_{15}-D_8 while the low bank is selected for A_0=0 and $\overline{BHE}$=1. Neither low bank nor high bank would selected for A_0= 1 and $\overline{BHE}$ = 1.

Fig. 12.4 shows how the total address space (1MB) of 8086 is physically implemented by segregating it into low and high banks. It also shows that $\overline{CS}$ signal of the high bank is connected to $\overline{BHE}$ while the $\overline{CS}$ signal of the low bank is connected to A_0.

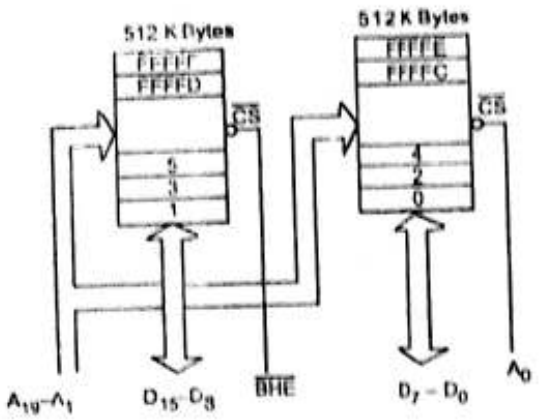


Fig.12.4: Selection of high and low banks of 8086

11. Show the profiles of low and high order memory banks.

Ans. The low and high order memory banks correspond to even and odd banks respectively.

The $\overline{CS}$ signal of low order memory bank is selected when $\overline{CS}$ = 0. Since A_0 (lowest address bus line) is connected to $\overline{CS}$, hence A_0 must have be to low for the low order bank to be selected. That is why the low order bank corresponds to even bank. Similarly the high order bank is selected when A_0 = 1. Hence, the higher order bank is called odd bank.

The profile of the low and high order banks are shown below in Fig. 12.5

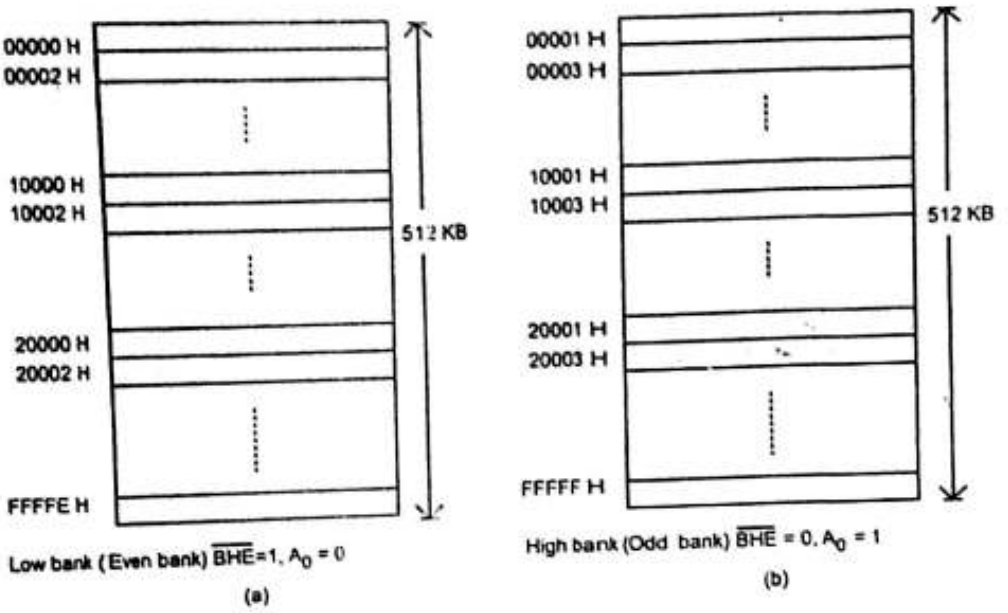


Fig. 12.5: (a) Low or even bank (b) High or odd bank

12. Draw the diagrams of (a) even-addressed byte transfer (b) odd-addressed byte transfer (c) even-addressed word transfer and (d) odd-addressed word transfer.

Ans. Fig. 12.6 shows the above four cases. A, B are representing the addresses while (A), (B) represent the content of address locations A and B respectively.

Figures (a) and (b) correspond to byte transfers for even and odd-addressed memory locations respectively. The shaded memory location indicates that the content of that particular memory location comes out either via higher byte data bus (D_{15}–D_8) or lower byte data bus (D_7–D_0) respectively.

Figures (a), (b) and (c) complete the data transfer in one bus cycle only.

For Figure (a), $\overline{BHE}$ = 1, A_0 = 0

For Figure (b), $\overline{BHE}$ = 0, A_0 = 1

For Figure (c), $\overline{BHE}$ = 0, A_0 = 0

Figure (d) corresponds to an odd-addressed word transfer and it takes two bus cycles to complete this transfer.

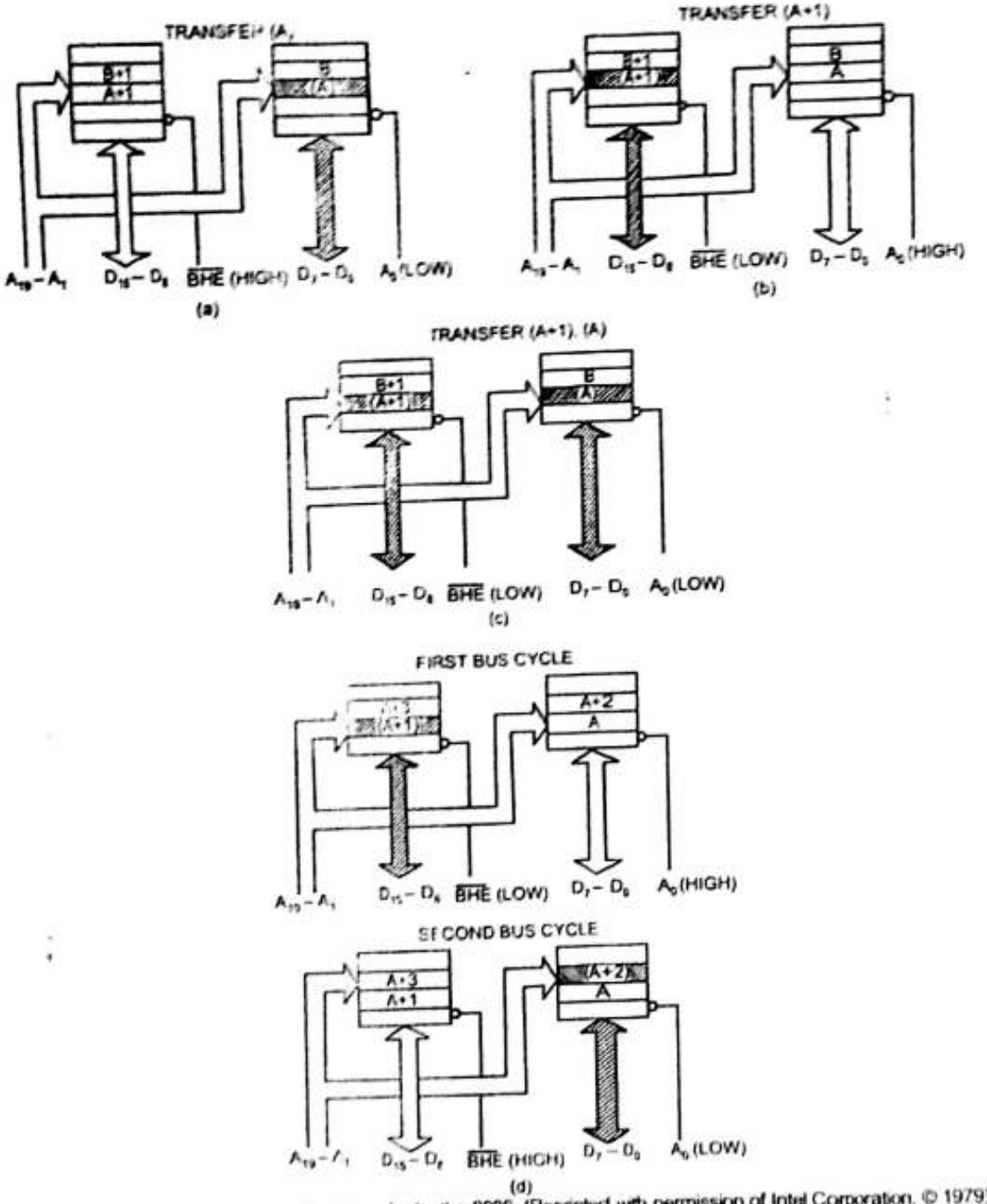


Fig. 12.8: (a) Even-addressed byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(b) Odd-addressed byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(c) Even-addressed word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(d) Odd-addressed word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)

This odd addressed word is an unaligned one and the LSB of the address is in the high memory bank

The odd byte of the word is at address location A + 1 and is selected by making $\overline{BHE} = 0$ and A_0. Thus in the first bus cycle, data to transferred on $D_{15} - D_8$.

In the second bus cycle, 8086 automatically increments the address. Hence A_0 becomes 0, representing even address A + 2. This is in the low bank and is accessed by making $\overline{BHE} = 1$ and $A_0 = 0$.

15. Which pins identify the segment registers used for 20-bit physical address generation?

Ans. Pins A_{18} and A_{17} become S_3 and S_4 from the second bus cycle. This 2-bit combination of S_3 and S_4 indicate the segment register used for physical address generation and is shown in Table 12.2.

Table 12.2 Identifying the segment register used for 20-bit physical address generation

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code/none
1	1	Data

The two status codes are output both in the maximum and minimum mode.

14. What is the maximum size of the memory that can be accessed by 8086?

Ans. The two status codes S_4 and S_3 together point to the segment register used for 20-bit physical address generation and can be examined by external circuitry to enable separate 1 MB address space for each of CS, ES, DS, and SS. This would enable memory address to be expanded to a maximum of 4 MB for 8086 µP.

15. Draw the Read and Write bus cycles for 8086 µP in Minimum mode.

Ans. Fig. 12.7 shows the Read and Write bus cycles for 8086 µP in the Minimum mode.

The bus cycle consists of 4T states. ALE signal stays high for T_1 state at the end of which it goes low which is utilised by latches to latch the address. Hence, during $T_2 - T_4$ states, $AD_{15} - AD_0$ lines act as data lines. The $M/\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals can be combined to generate individual $\overline{IOR}$, $\overline{IOW}$ and $\overline{MEMR}$, $\overline{MEMW}$ signals.

The Read and Write cycles show that data are made available during T_3 and T_2 states respectively.

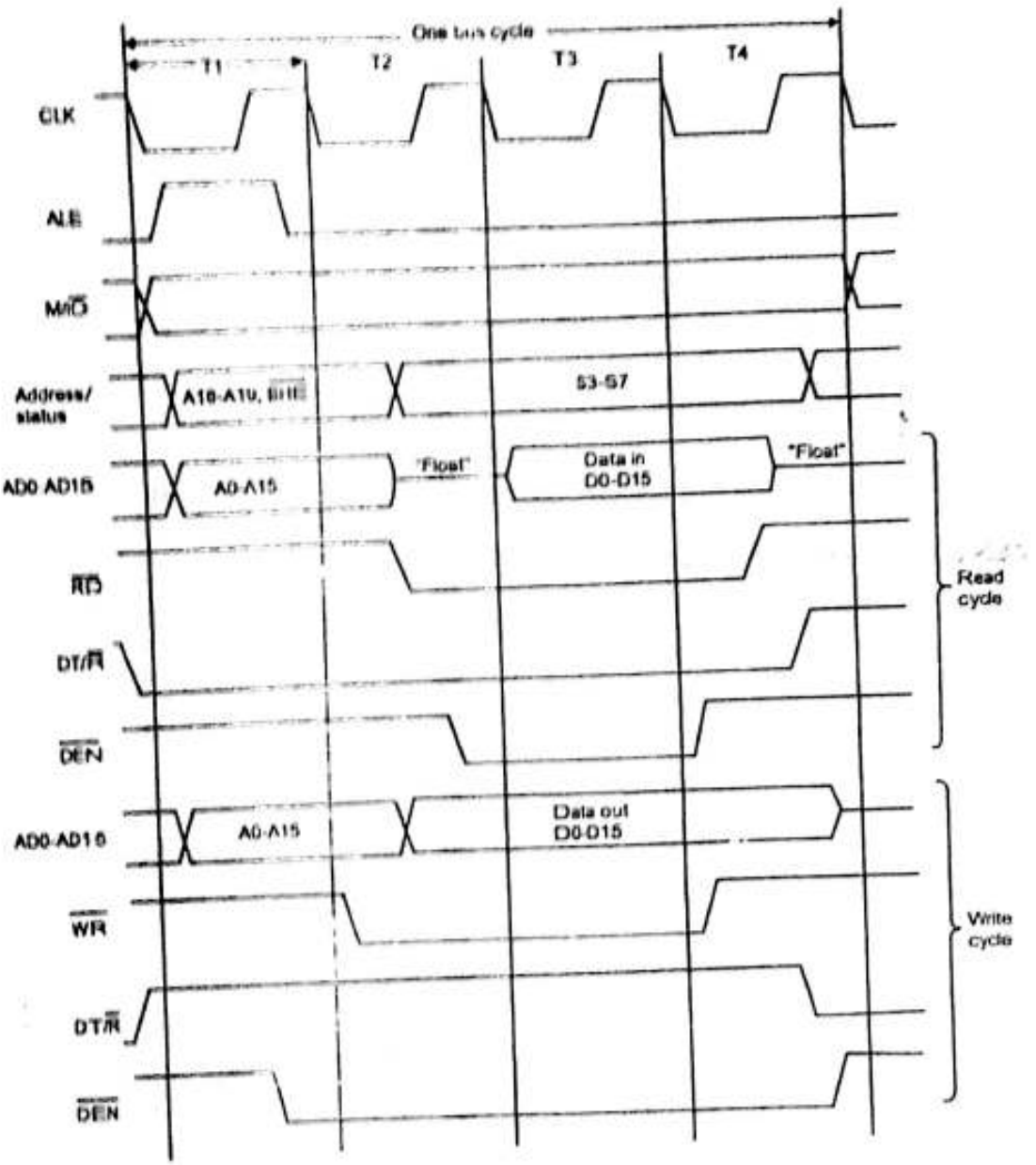


Fig.12.7: 8086 microprocessor read and write bus cycles The address lines are valid during the T1 state but become the data lines and status indicators during T2–T4

8086 Interrupt (Review Sheet 8)

1- How many interrupts can be implemented using 8086 μP?
A total of 256 interrupts can be implemented using 8086 μP.

2. Mention and tabulate the different types of interrupts that 8086 can implement.
8086 μP can implement seven different types of interrupts.
- NMI and INTR are external interrupts implemented via *Hardware*.
- INT n, INTO and INT3 (breakpoint instruction) are software interrupts implemented through *Program*.
- The 'divide-by-0' and 'Single-step' are interrupts *initiated by CPU*.

3- How many bytes are needed to store the starting addresses of ISS for 8086
8086 μP can implement 256 different interrupts. To store the starting address of a single ISS (Interrupt Service Subroutine), four bytes of memory space are required two bytes to store the value of CS and two bytes to store the IP value. Thus to store the starting address of 256 ISS, in all 256 × 4 = 1024 bytes = 1 KB will be required.

4-Indicate the number of memory spaces needed in stack when an interrupt occurs.
When an interrupt occurs, before moving over to starting address of the corresponding ISS, the following are pushed into the stack: the contents of the flag register, CS and IP. Since each one of the three are 2 bytes, hence a total of 6 bytes of memory space is needed in the stack to accommodate the flag register, CS and IP contents.

5- Distinguish between the two hardware interrupts of 8086.
The distinction between the two hardware interrupts of 8086 are as follows, shown in the following table

NMI	INTR
1. Non-maskable type.	1. Maskable type.
2. Higher priority.	2. Lower priority.
3. Edge triggered interrupt initiated on Low to High transition.	3. Level triggered interrupt.
4. Must remain high for more than 2 CLK cycles.	4. Sampled during last CLK cycle of each instruction.
5. The rising edge of NMI input is latched on-chip and is serviced at the end of current instruction.	5. No latching. Must stay high until acknowledged by CPU.
6. No acknowledgement.	6. Acknowledged by INTA output signal.

6- What are meant by interrupt pointer and interrupt pointer table?

The starting address of an ISS in the 1 KB memory space is known as the interrupt pointer or interrupt vector corresponding to that interrupt.

The 1 KB memory space needed to store the starting addresses of all the 256 ISS is called the interrupt pointer table.

7- Write down the steps, sequentially carried out by the systems when an interrupt occurs.

When an interrupt occurs (hardware or software), the following things happen:

- The contents of flags register, CS and IP are pushed on to the stack.
- TF and IF are cleared which disable single step and INTR interrupts respectively.
- Program jumps to the starting address of ISS.
- At the end of ISS, when IRET is executed in the last line, the contents of flag register, CS and IP are popped out of the stack and placed in the respective registers.
- When the flags are restored, IF and TF get back their previous values.

8- Indicate two applications where NMI interrupt can be applied.

NMI is a non-maskable hardware interrupt, i.e., it cannot be masked or disabled. Hence, it is used for very important system exigencies like

- detection of power failure or
- detection of memory read error cases.

9- What way the INTO instruction is different from others?

The INTO instruction is different in that no type number is needed to be mentioned.

To explain the difference, for executing any INT instruction, type no. is needed, like

INT 10, INT 23, etc.

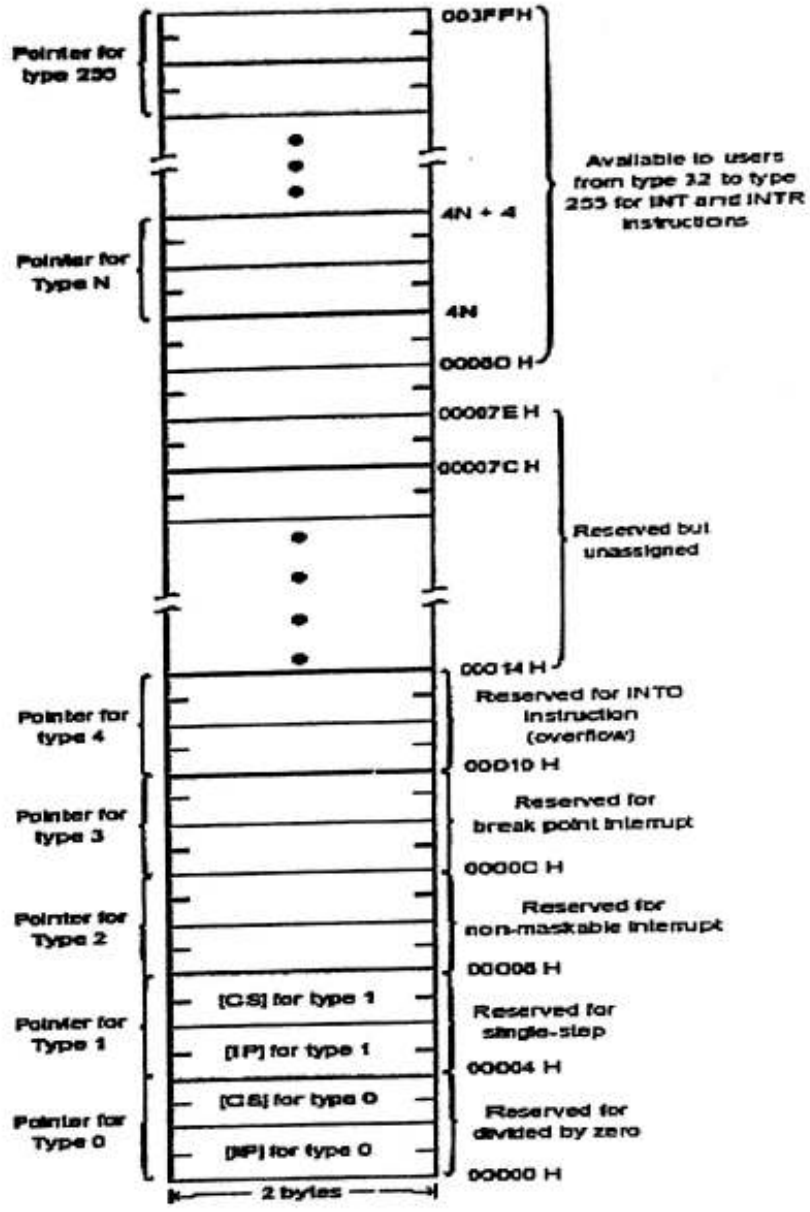
To explain further,

Opcode	Operand Object Type	Code Mnemonic	
INT		CD 23	INT 23 H (assuming Type 23 H is employed)
INTO	none	CE	INT

10- Draw and discuss the interrupt pointer table for 8086?

The 256 interrupt pointers are stored in memory locations starting from 00000 H to003FF H (1 KB memory space). The number assigned to an interrupt pointer is called the Type of the corresponding interrupt.

As for example, Type 0 interrupt, Type 1interrupt ... Type 255 interrupt. Type 0 interrupt has a memory address 00000 H, Type 1has a memory address 00004 H, while Type 255 has a memory address 003FF H. The first five pointers (Type 0 to Type 4) are dedicated pointers used for divide by zero, single step, NMI, break point and overflow interrupts respectively. The next 27 pointers (Type 5 to Type 31) are reserved pointers—reserved for some special interrupts. The remaining 224 interrupts—from Type 32 to Type 255 are available to the programmer for handling hardware and software interrupts.



11- Discuss the priority of interrupts of 8086.

8086 tests for the occurence of interrupts in the following hierarchical sequence:

- Internal interrupts (divide-by-0, single step, break point and overflow)
- Non-maskable interrupt—via NMI
- Software interrupts—via INTn
- External hardware interrupt—via INTR

Hence, internal interrupts belong to the highest priority group and internal hardware interrupts are the lowest priority group. Again, different interrupts are given different priorities by assigning a type number corresponding to each priority—starting from Type 0 (highest priority interrupt) to Type 255 (lowest priority interrupt). Thus, Type 40 interrupt is having more priority than Type 41 interrupt. If we presume that at any instant a Type 40 interrupt is in progress, then it can be interrupted by any software interrupt, the non-maskable interrupt, all internal interrupts or any external interrupt with a Type number less than 40.

12- Show the internal interrupts and their priorities.

The internal interrupts are: Divide-by-0, single step, break point and overflow corresponding to Type 0, Type 1, Type 3 and Type 4 interrupts respectively. Since a type with lesser number has higher priority than a type with more number, thus the mentioned internal interrupts can be arranged in a decreasing priority mode, with highest priority mentioned first: Divide-by-0, single step, break point, overflow.

13- Mention the addresses at which CS_{40} and IP_{40} corresponding to vector 40 would be stored in memory.

INT 40, for its storage, requires four memory locations—two for IP_{40} and two for CS_{40}.

The addresses are calculated as follows:

$4 \times 40 = 160_{10}$

$= 1010\ 0000$

$= A0\ H.$

Thus, IP_{40} is stored starting at 000A0 H and CS_{40} is stored starting at 000A2 H.

14- Discuss the Type 2 interrupt (non-maskable NMI interrupt)

Type 2 interrupt is the non-maskable NMI interrupt and is used for some emergency situations like power failure. When power fails, an external circuit detects this and sends an interrupt signal via NMI pin of 8086. The DC supply remains on for atleast 50 ms via capacitor banks so that the program and data remaining in RAM locations can be saved, which were being executed at the time of power failure.

The 8086 Microprocessor

1. Draw the pin diagram of 8086.

Ans. There would be two pin diagrams—one for MIN mode and the other for MAX mode of 8086, shown in Figs. 11.1 and 11.2 respectively. The pins that differ with each other in the two modes are from pin-24 to pin-31 (total 8 pins).

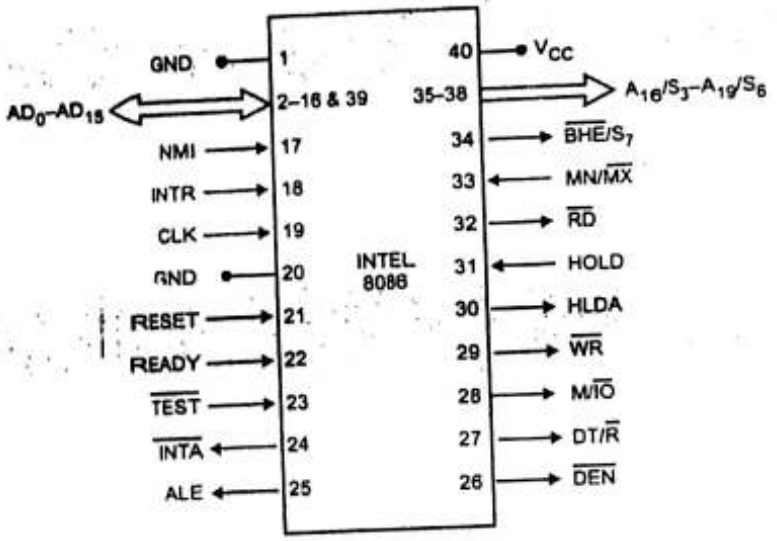


Fig. 11.1: Signals of Intel 8086 for minimum mode of operation

2. What is the technology used in 8086 μP?

Ans. It is manufactured using high performance metal-oxide semiconductor (HMOS) technology. It has approximately 29,000 transistors and housed in a 40-pin DIP package.

3. Mention and explain the modes in which 8086 can operate.

Ans. 8086 μP can operate in two modes—MIN mode and MAX mode.

When MN/$\overline{MX}$ pin is high, it operates in MIN mode and when low, 8086 operates in MAX mode.

For a small system in which only one 8086 microprocessor is employed as a CPU, the system operates in MIN mode (Uniprocessor). While if more than one 8086 operate in a system then it is said to operate in MAX mode (Multiprocessor).

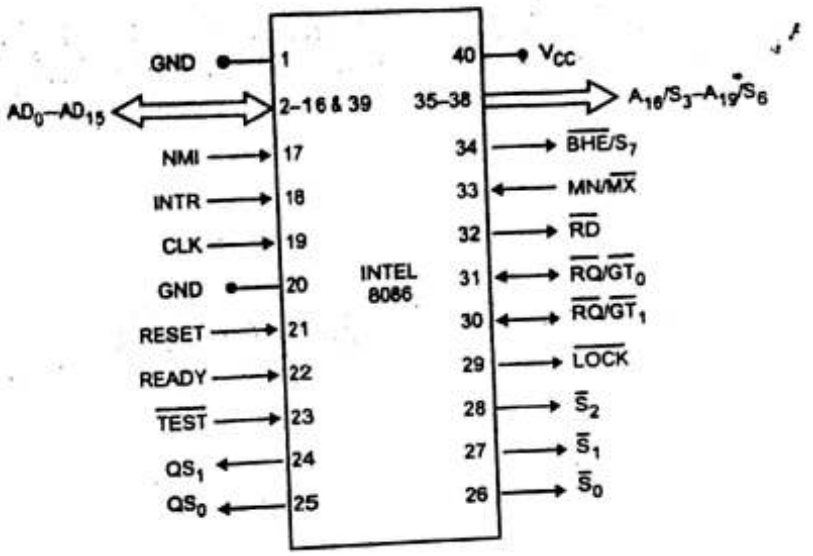


Fig. 11.2: Signals of intel 8086 for maximum mode of operation

The bus controller IC (8288) generates the control signals in case of MAX mode, while in MIN mode CPU issues the control signals required by memory and I/O devices.

4. Distinguish between the lower sixteen address lines from the upper four.

Ans. Both the lower sixteen address lines ($AD_0 - AD_{15}$) and the upper four address lines ($A_{16}/S_3 - A_{19}/S_6$) are multiplexed.

During T_1, the lower sixteen lines carry address ($A_0 - A_{15}$), while during T_2, T_3 and T_4 they carry data.

Similarly during T_1, the upper four lines carry address ($A_{16} - A_{19}$), while during T_2, T_3 and T_4, they carry status signals.

5. In how many modes the minimum-mode signal can be divided?

Ans. In the MIN mode, the signals can be divided into the following basic groups: address/data bus, status, control, interrupt and DMA.

6. Tabulate the common signals, Minimum mode signals and Maximum mode signals. Also mention their functions and types.

Ans. Table 11.1 shows the common signals, Minimum mode signals and the Maximum mode signals, along with the functions of each and their types.

Table 11.1 : (a) Signals common to both minimum and maximum mode, (b) Unique minimum-mode signals, (c) Unique maximum-mode signals for 8086.

Common signals

Name	Function	Type
AD15-AD0	Address/data bus	Bidirectional, 3-state
A19/S6-A16/S3	Address/status	Output, 3-state
MN/$\overline{MX}$	Minimum/maximum Mode control	Input
$\overline{RD}$	Read control	Output, 3-state
$\overline{TEST}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Nonmasakable interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{cc}	+5V	Input
GND	Ground	

(a)

Minimum mode signals (MN/$\overline{MX}$ =V_{cc})

Name	Function	Type
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{WR}$	Write control	Output, 3-state
M/$\overline{IO}$	IO/memory control	Output, 3-state
DT/$\overline{R}$	Data transmit/receive	Output, 3-state
$\overline{DEN}$	Data enable	Output, 3-state
ALE	Address latch enable	Output
$\overline{INTA}$	Interrupt acknowledge	Output

(b)

Maximum mode signals (MN/$\overline{MX}$ =GND)

Name	Function	Type
$\overline{RQ}$/$\overline{GT1,0}$	Request/grant bus access control	Bidirectional
$\overline{LOCK}$	Bus priority lock control	Output, 3-state
$\overline{S2}$ - $\overline{S0}$	Bus cycle status	Output, 3-state
QS1, QS0	Instruction queue status	Output

(c)

7. Mention the different varieties of 8086 and their corresponding speeds.

Ans. The following shows the different varieties of 8086 available and their corresponding speeds.

Types	Speeds
8086	5 MHz
8086–1	10 MHz
8086–2	8 MHz

8. Mention (a) the address capability of 8086 and (b) how many I/O lines can be accessed by 8086.

Ans. 8086 addresses via its A_0–A_{19} address lines. Hence it can address 2^{20} = 1MB memory.
Address lines A_0 to A_{15} are used for accessing I/O's. Thus, 8086 can access 2^{16} = 64 KB of I/O's.

9. What is meant by microarchitecture of 8086?

Ans. The individual building blocks of 8086 that, as a whole, implement the software and hardware architecture of 8086. Because of incorporation of additional features being necessitated by higher performance, the microarchitecture of 8086 or for that matter any microprocessor family, evolves over time.

10. Draw and discuss the architecture of 8086. Mention the jobs performed by BIU and EU.

Ans. The architecture of 8086 is shown below in Fig. 11.3. It has got two separate functional units—Bus Interface Unit (BIU) and Execution Unit (EU).

8086 architecture employs parallel processing—i.e., both the units (BIU and EU) work at the same time. This is *Unlike* 8085 in which *Sequential* fetch and execute operations take place. Thus in case of 8086, efficient use of system bus takes place and higher performance (because of reduced instruction time) is ensured.

- BIU has segment registers, instruction pointer, address generation and bus control logic block, instruction queue while the EU has general purpose registers, ALU, control unit, instruction register, flag (or status) register.

The main jobs performed by BIU are:

- BIU is the 8086's interface to the outside world, i.e., all *External* bus operations are done by BIU.
- It does the job of instruction fetching, reading/writing of data/operands for memory and also the inputting/outputting of data for peripheral devices.
- It does the job of filling the instruction queue.
- Does the job of address generation.

The main jobs performed by the execution unit are:

- Decoding/execution of instructions.
- It accepts instructions from the output end of instruction queue (residing in BIU) and data from the general purpose registers or memory.
- It generates operand addresses when necessary, hands them over to BIU requesting it (BIU) to perform read or write cycle to memory or I/O devices.
- EU tests the status of flags in the control register and updates them when executing instructions.
- EU waits for instructions from the instruction queue, when it is empty.
- EU has no connection to the system buses.

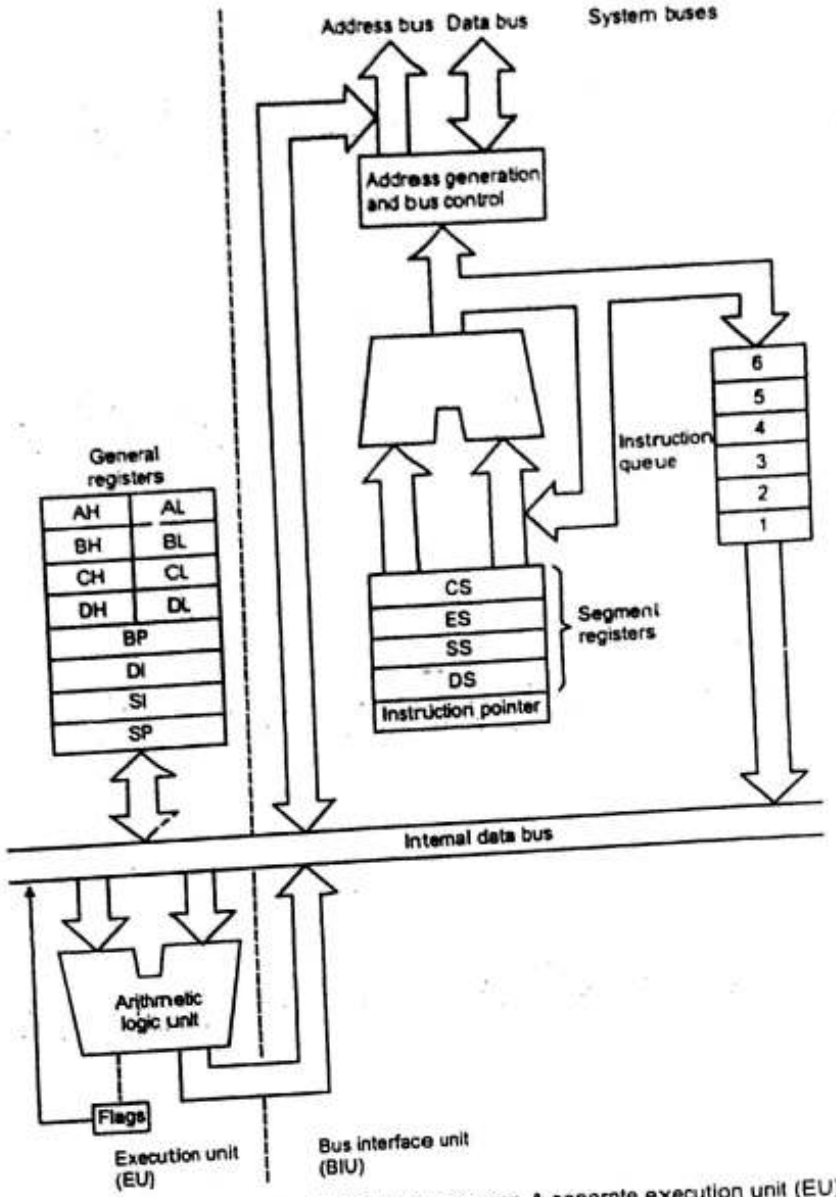


Fig. 11.3: CPU model for the 8086 microprocessor. A separate execution unit (EU) and bus interface unit (BIU) are provided.

11. Explain the operations of instructions queue residing in BIU.

Ans. The instruction queue is 6-bytes in length, operates on FIFO basis, and receives the instruction codes from memory. BIU fetches the instructions meant for the queue ahead of time from memory. In case of JUMP and CALL instructions, the queue is dumped and newly formed from the new address.

respectively. For such cases, contents of SI are added to contents of DS register to get the actual source address of data, while the contents of DI are added to the contents of ES to get the actual destination address of data.

SP and BP stand for stack pointer and base pointer with SP containing the offset address or the stack top address. The actual stack address is computed by adding the contents of SP and SS.

Data area(s) may exist in stack. To access such data area in stack segment, BP register is used which contains the offset address. BP register is also used as a general purpose register.

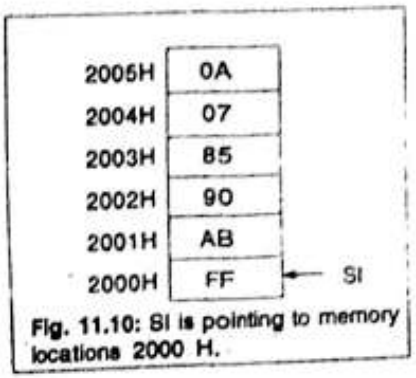


Fig. 11.10: SI is pointing to memory locations 2000 H.

Instruction pointer (IP) is also included in the index and pointers group. IP points to the offset instead of the actual address of the next instruction to be fetched (from the current code segment) in BIU. IP resides in BIU but cannot be programmed by the programmer.

19. Describe in brief the four segment registers.

Ans. The four segment registers are CS, DS, ES and SS—standing for code segment register, data segment register, extra segment register and stack segment register respectively. When a particular memory is being read or written into, the corresponding memory address is determined by the content of one of these four segment registers in conjunction with their offset addresses.

The contents of these registers can be changed so that the program may jump from one active code segment to another one.

The use of these segment registers will be more apparent in memory segmentation schemes.

20. Discuss A_{16}/S_3—A_{19}/S_6 Signals of 8086.

Ans. These are time multiplexed signals. During T_1, they represent $A_{19} - A_{16}$ address lines. During I/O operations, these lines remain low. During T_2–T_4, they carry status signals.

S_4 and S_3 (during T_2 to T_4) identify the segment register employed for 20-bit physical address generation.

Status signal S_5 (during T_2 to T_4) represents interrupt enable status. This is updated at the beginning of each clock cycle.

Status signal S_6 remains low during T_2 to T_4.

21. Discuss $\overline{BHE}/S_7$ signal.

Ans. During T_1, this becomes bus high enable signal and remains low while during T_2 to T_4 it acts as a status signal S_7 and remains high during this time.

During T_1, when $\overline{BHE}$ signal is active, i.e., remains low, it is used as a chip select signal on the higher byte of data bus—i.e., D_{15}–D_8.

Table 11.2 shows $\overline{BHE}$ and A_0 signals determine one of the three possible references to memory.

Table 11.2: Status of $\overline{BHE}$ and A_0 identify memory references

$\overline{BHE}$	A_0	Word/byte access
0	0	Both banks active, 16-bit word transfer on $AD_{15} - AD_0$
0	1	Only high bank active, upper byte from/to odd address on $AD_{15} - AD_8$
1	0	Only low bank active, lower byte from/to even address $AD_7 - AD_0$
1	1	No bank active

22. Discuss the Reset pin of 8086.

Ans. Reset is an active high input signal and must be active for at least 4 CLK cycles to be accepted by 8086. This signal is internally synchronised and execution starts only after Reset returns to low value.

For proper initialisation, Reset pulse must *not* be applied before 50μS of 'power on' of the circuit. During Reset state, all three buses are tristated and ALE and HLDA are driven low.

During resetting, all internal register contents are set to 0000 H, but CS is set to FO00 H and IP to FFF0 H. Thus execution starts from physical address FFFF0 H. Thus EPROM in 8086 is interfaced so as to have the physical memory location forms FFFF0 H to FFFFF H, i.e., at the end of the map.

23. Discuss the two pins (a) DT/$\overline{R}$ and (b) $\overline{DEN}$.

Ans. (a) DT/$\overline{R}$ is an output pin which decides the directions of data flow through the transreceivers (bidirectional buffers).

When the processor sends out data, this signal is 1 while when it receives data, the signal status is 0.

(b) $\overline{DEN}$ stands for data enable. It is an active low signal and indicates the availability of data over the address/data lines. This signal enables the transreceivers to separate data from the multiplexed address/data signal. It is active from the middle of T_2 until the middle of T_4.

Both DT/$\overline{R}$ and $\overline{DEN}$ are tristated during 'hold acknowledge'.

24. Elaborate the functions of the pins $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.

Ans. These three are output status signals in the MAX mode, indicating the type of operation carried out by the processor.

The signals become active during T_4 of the previous cycle and remain active during T_1 and T_2 of the current cycle. They return to the passive state during T_3 of the current bus cycle so that they may again become active for the next bus cycle during T_4. Table 11.3 shows the different bus cycles of 8086 for different combinations of these three signals.

Table 11.3: Bus status codes

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CPU cycles
0	0	0	Interrupt acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	HALT
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

25. Explain the $\overline{\text{LOCK}}$ signal.

Ans. It is an active low output signal and is activated by LOCK prefix instruction and remains active until the completion of the next instruction. It floats to tri-state during hold acknowledge when $\overline{\text{LOCK}}$ signal is low, all interrupts get masked and HOLD request is not granted. $\overline{\text{LOCK}}$ signal is used by the processor to prevent other devices from accessing the system control bus. This symbol is used when CPU is executing some critical instructions and through this signal other devices are informed that they should not issue HOLD signal to 8086.

26. Explain the $\overline{\text{TEST}}$ signal.

Ans. It is an active low input signal. Normally the BUSY pin (output) of 8087 NDP is connected to the $\overline{\text{TEST}}$ input pin of 8086. When maths co-processor 8087 is busy executing some instructions, it pulls its BUSY signal high. Thus the $\overline{\text{TEST}}$ signal of 8086 is consequently high, and it (8086) is made to WAIT until the BUSY signal goes low. When 8087 completes its instruction executions, BUSY signal becomes low. Thus the $\overline{\text{TEST}}$ input of 8086 becomes low also and then only 8086 goes in for execution of its program.

27. Show how demultiplexing of address/data bus is done and also show the availability of address/data during read/write cycles.

Ans. The demultiplexing of lower 2-bytes of address/data bus (AD_0–AD_{15}) is done by 8282/8283 octal latch with 8282 providing non-inverting outputs while 8283 gives out inverted outputs. The chip outputs are also buffered so that more drive is available at their outputs.

A D latch is central to the demultiplexing operations of these latches. During T_1 when ALE is high, the latch is transparent and the output of latch is 'A' (address) only. At the end of T_1, ALE has a high to low transition which latches the address available at the D input of the latch, so that address is continued to be available from the Q output of the latch (i.e., whole of T_1 to T_4 states).

It is to be noted that memory and I/O devices do not access the data bus until the beginning of T_2, thus the 'data' is a 'don't care' till the end of T_1. This is shown in Fig. 11.11 and the timing diagram shows the availability of data for read and write cycles.

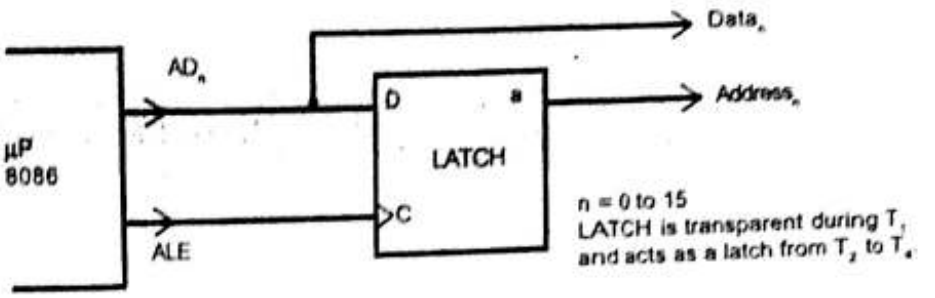


Fig.11.11: Demultiplexing the 8086 address/data bus

28. Discuss the Instruction Pointer (IP) of 8086.

Ans. Functionally, IP plays the part of Program Counter (PC) in 8085. But the difference is that IP holds the offset of the next word of the instruction code instead of the actual address (as in PC).

IP along with CS (code segment) register content provide the 20-bit physical (or real) address needed to access the memory. Thus CS:IP denotes the value of the memory address of the next code (to be fetched from memory).

Content of IP gets incremented by 2 because each time a word of code is fetched from memory.

29. Indicate the data types that can be handled by 8086 μP.

Ans. The types of data formats that can be handled by 8086 fall under the following categories:

- Unsigned or signed integer numbers—both byte-wide and word-wide.
- BCD numbers—both in packed or unpacked form.
- ASCII coded data. ASCII numbers are stored one number per byte.

30. Compare 8086 and 8088 microprocessors.

Ans. The Comparison between the two is tabulated below in Table 11.4.

Table 11.4: Comparison of 8086 and 8088

8086	8088
1. 2-byte data width, obtained by demultiplexing AD_0–AD_{15}.	1. 1-byte data width, obtained by demultiplexing AD_0–AD_7.
2. In MIN mode, pin-28 is assigned the signal M/$\overline{IO}$.	2. In MIN mode, pin-28 is assigned the signal IO/$\overline{M}$.
3. A 6-byte instruction queue.	3. A 4-byte instruction queue.
4. To access higher byte, $\overline{BHE}$ signal is used.	4. No such signal required, since data width is 1-byte only.
5. BIU dissimilar, but EU similar to 8088. Program instructions identical to 8088.	5. BIU dissimilar, but EU similar to 8086. Program instructions identical to 8086.
6. Program fetching from memory done only when 2-bytes are empty in queue.	6. Program fetching from memory done as soon as a byte is free in queue.
7. Pin-34 is $\overline{BHE}$/S_7. During T_1, $\overline{BHE}$ is used to enable data on D_8–D_{15}. During T_2–T_4, status of this pin is 0. In MAX mode, 8087 monitors this pin to identify the CPU—8086 or 8088? Accordingly it sets its queue length to 6 or 4 respectively.	7. Pin-34 is $\overline{SS_0}$. It acts as $\overline{S_0}$ in the MIN mode. In MAX mode $\overline{SS_0}$ = 1 always.